Technical Manual PD1-C

Fieldbus: Modbus RTU

For use with the following variants:

PD1-C281S15-E-20-5, PD1-C281S15-E-65-5, PD1-C281S15-E-OF-5, PD1-C281L15-E-20-5, PD1-C281L15-E-65-5, PD1-C281L15-E-OF-5

Valid with firmware version FIR-v2425
and since hardware version W002

Technical Manual Version: 1.1.0

Contents

1 Introduction

The *PD1-C* is a stepper motor with integrated controller. The integrated absolute encoder makes immediate operation possible in *closed loop* mode without homing.

This manual describes the functions of the controller and the available operating modes. It also shows how you can address and program the controller via the communication interface.

You can find further information on the product on us.nanotec.com.

1.1 Version information

Manual version	Date	Changes	Firmware version	Hardware version
1.0.0	10/2023	Edition	B1048823	W002
1.0.1	11/2023	Minor corrections	B1048823	W002
1.0.2	12/2023	Pin arrangement for IP65 variants corrected.	B1048823	W002
1.1.0	07/2024	New Firmware: *Slow Speed Mode* surrently not supported. Minor corrections.	FIR-v2425	W002

1.2 Copyright, marking and contact

Nanotec Electronic GmbH & Co. KG

Kapellenstraße 6

85622 Feldkirchen

Germany

Phone: +49 89 900 686-0

Fax: +49 (89) 900 686-50

us.nanotec.com

1.3 Intended use

The *PD1-C* motor with integrated controller is used as a component of drive systems in a range of industrial applications.

Use the product as intended within the limits defined in the technical data (in particular, see) and the approved Environmental conditions.

Under no circumstances may this Nanotec product be integrated as a safety component in a product or system. All products containing a component manufactured by Nanotec must, upon delivery to the end user, be provided with corresponding warning notices including instructions for safe use and safe operation. All warning notices provided by Nanotec must be passed on directly to the end user.

1.4 Target group and qualification

The product and this documentation are directed towards technically trained specialists staff such as:

- Development engineers
- Plant engineers
- Installers/service personnel
- Application engineers

Only specialists may install, program and commission the product. Specialist staff are persons who

- have appropriate training and experience in working with motors and their controller,
- are familiar with and understand the content of this technical manual,
- know the applicable regulations.

1.5 Warranty and disclaimer

Nanotec shall not be liable for damage and malfunctions attributable to installation errors, failure to observe this document or improper repair. The plant engineer, operating company and user shall be responsible for the selection, operation and use of our products. Nanotec shall not take responsibility for integration of the product in the end system. The general terms and conditions listed at www.nanotec.de shall apply. **Note:** Conversion/modification of the product is prohibited.

1.6 EU directives for product safety

The following EU directives were observed:

- RoHS directive (2011/65/EU, 2015/863/EU)
- EMC directive (2014/30/EU)

NOTICE

For product variants without closed housing (*PD1-C…-…-**OF**-…*), no EMC tests were performed. Perform a risk assessment for the entire machine/system to identify possible risks due to electromagnetic interference and initiate suitable protection measures if necessary.

1.7 Other applicable regulations

In addition to this technical manual, the following regulations are to be observed:

- Accident-prevention regulations
- Local regulations on occupational safety

1.8 Used icons

All notices are in the same format. The degree of the hazard is divided into the following classes.

CAUTION!

The CAUTION notice indicates a possibly dangerous situation.

Failure to observe the notice **may** result in moderately severe injuries.

► Describes how you can avoid the dangerous situation.

NOTICE

Indicates a possible incorrect operation of the product.

Failure to observe the notice may result in damage to this or other products.

▶ Describes how you can avoid the incorrect operation.

TIP

Shows a tip for the application or task.

1.9 Emphasis in the text

The following conventions are used in the document:

Underlined text indicates cross references and hyperlinks:

- The following bits in object 6041_h (statusword) have a special function:
- A list of available system calls can be found in chapter NanoJ functions in the NanoJ program.

Text set in *italics* marks named objects:

- Read the *installation manual*.
- Use the *Plug & Drive Studio* software to perform the auto setup.
- For software: You can find the corresponding information in the *Operation* tab.
- For hardware: Use the *ON/OFF* switch to switch the device on.

A text set in `Courier` marks a code section or programming command:

- The line with the `od_write(0x6040, 0x00, 5 );` command has no effect.
- The NMT message is structured as follows: `000 | 81 2A`

A text in "quotation marks" marks user input:

- Start the NanoJ program by writing object 2300_h, bit 0 = "1".
- If a holding torque is already needed in this state, the value "1" must be written in 3212_h:01_h.

1.10 Numerical values

Numerical values are generally specified in decimal notation. The use of hexadecimal notation is indicated by a subscript *h* at the end of the number.

The objects in the object dictionary are written with index and subindex as follows: `<Index>:<Subindex>`

Both the index as well as the subindex are specified in hexadecimal notation. If no subindex is listed, the subindex is 00_h.

Example: Subindex 5 of object 1003_h is addressed with `1003`$_h$`:05`$_h$, subindex 00 of object 6040_h with `6040`$_h$.

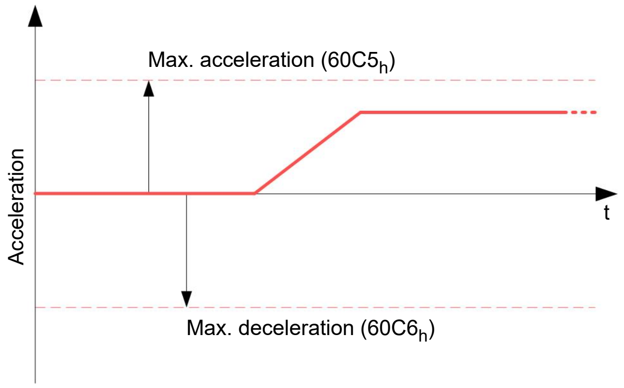
1.11 Bits

The numbering of individual bits in an object always begins with the LSB (bit number 0). See the following figure, which uses data type *UNSIGNED8* as an example.

	MSB							LSB	
Bit Nummer	7	6	5	4	3	2	1	0	
Bits	0	1	0	1	0	1	0	1	≙ 55_{hex} ≙ 85_{dec}

1.12 Counting direction (arrows)

In figures, the counting direction is always in the direction of an arrow. Objects `60C5h` and `60C6h` depicted as examples in the following figure are both specified as positive.



2 Safety and warning notices

CAUTION!

Risk of burning from hot surfaces!

The motor can become very hot during operation. If touched, this could result in burns.

► During use, make certain that the environmental conditions are ensured and that operation takes place within the limits defined by the technical data.

► Install the motor in such a way that heat dissipation and passive cooling are possible.

► After switching off, wait until all components have cooled before you touch them.

NOTICE

Damage to the controller!

Changing the wiring during operation may damage the controller.

► Only change the wiring in a de-energized state. After switching off, wait until the capacitors have discharged.

NOTICE

Damage to the controller due to excitation voltage of the motor!

Voltage peaks during operation may damage the controller.

► Install suitable circuits (e.g., charging capacitor) that reduce voltage peaks.

NOTICE

Damage to the electronics through improper handling of ESD-sensitive components!

The device contains components that are sensitive to electrostatic discharge. Improper handling can damage the device.

► Observe the basic principles of ESD protection when handling the device.

NOTICE

Damage to the electronics if the supply voltage is connected with reversed polarity!

► Install a line protection device (fuse) in the supply line.

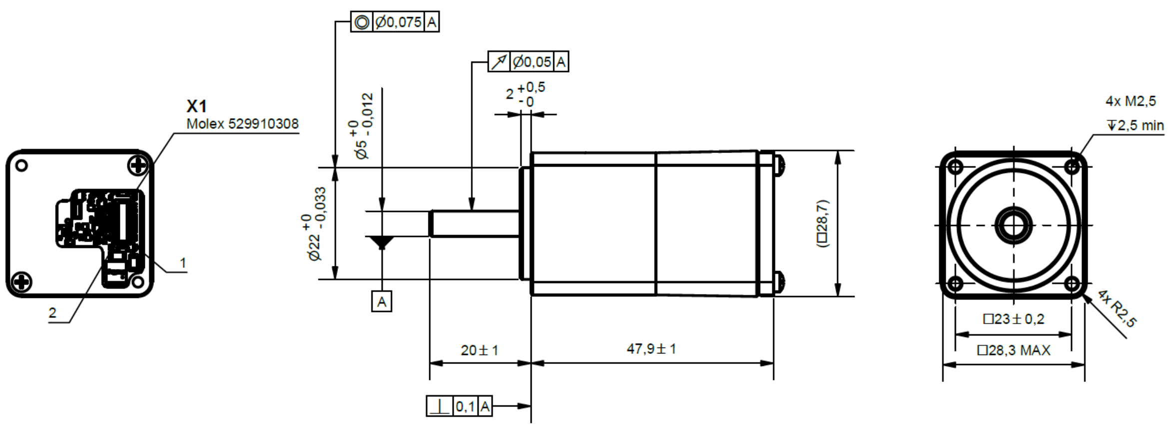
3 Technical details and pin assignment

3.1 Environmental conditions

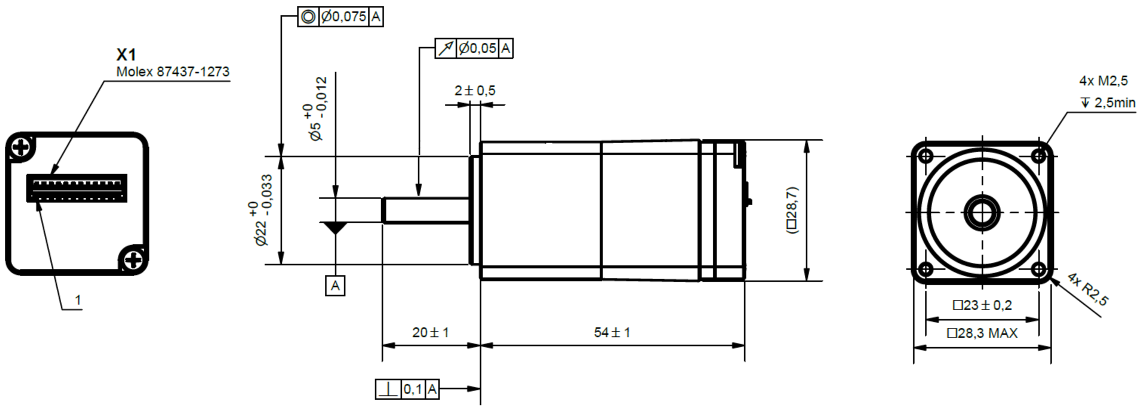
Environmental condition	Value
Protection class	■ *PD1-C…-…-**OF**-…*: No IP protection ■ *PD1-C…-…-**20**-…*: IP20 ■ *PD1-C…-…-**65**-…*: IP65 (except for shaft output)
Ambient temperature (operation)	-10 … +40°C
Ambient temperature (storage and transport)	-25 … +85°C
Relative humidity (operation), non-condensing	0 … 85%
Relative humidity (storage and transport), non-condensing	0 … 85%
Absolute humidity (storage and transport), non-condensing	30 g/m^3
Max. altitude of site above *sea level* (without drop in performance in operation)	1500 m
Max. altitude of site above *sea level* (storage and transport)	3000 m

3.2 Dimensioned drawings

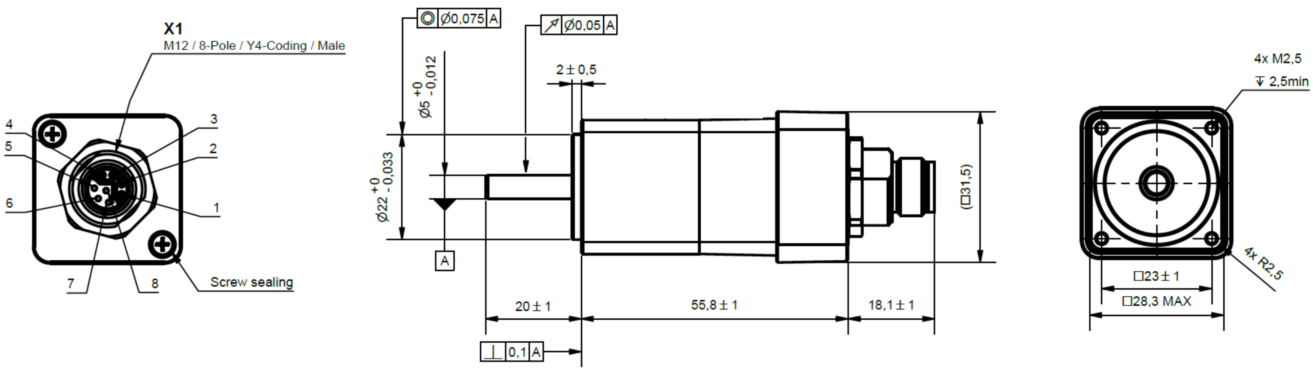
PD1-C281S15-E-OF-…



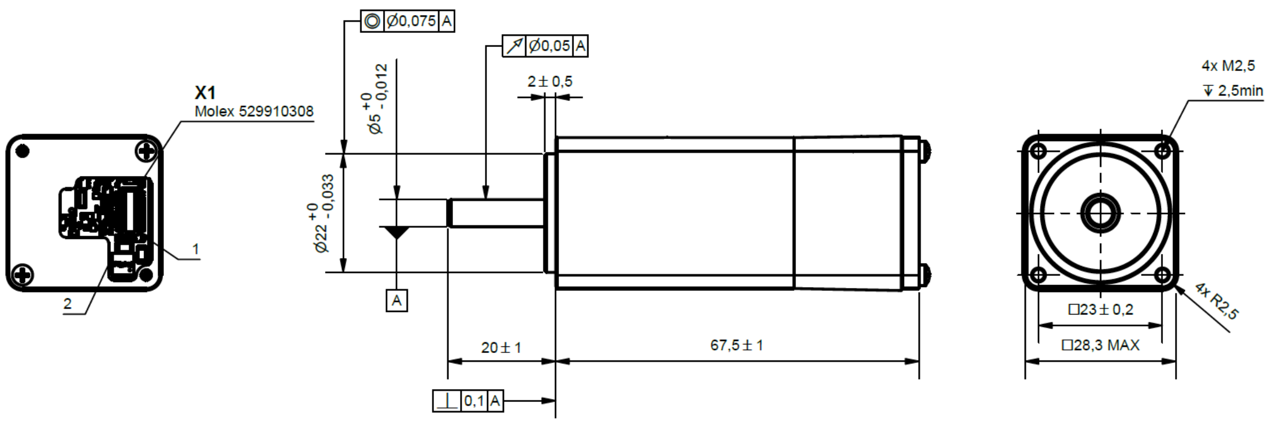
PD1-C281S15-E-20-…



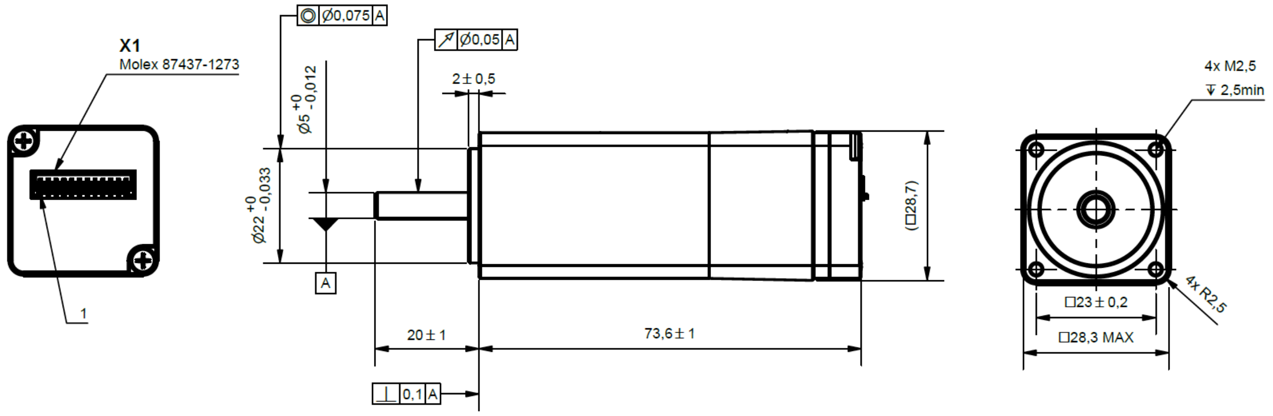
PD1-C281S15-E-65-…



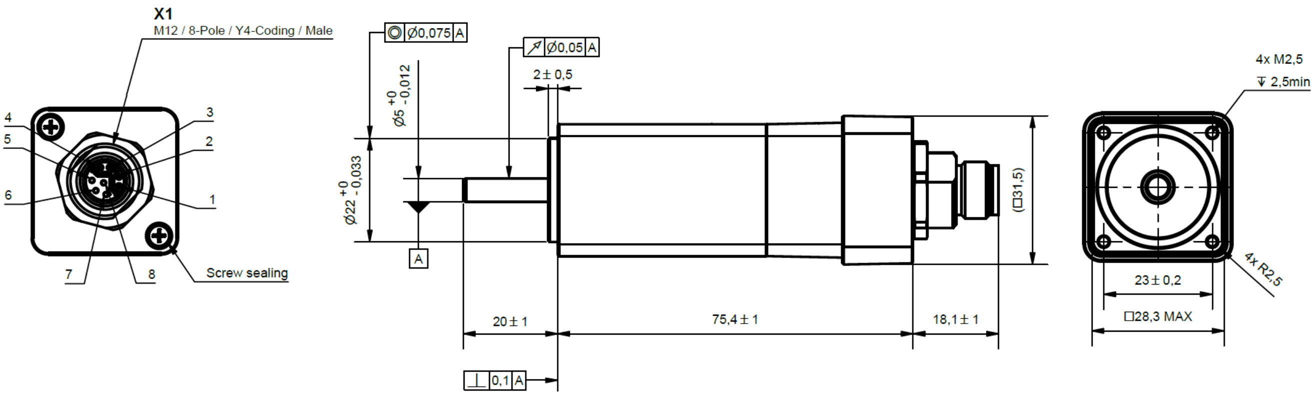
PD1-C281L15-E-OF-…



PD1-C281L15-E-20-…



PD1-C281L15-E-65-…



3.3 Electrical properties and technical data

Property	Description / value
Operating voltage	12 V DC to 30 V DC
Rated current	1.5 A_{rms}
Peak current	3 A_{rms} for max. 3 seconds **Note:** To avoid voltage drops at peak current that would cause an under-voltage error at voltages near the lower limit (below 15 V), connect a capacitor of at least 4700 µF / 50 V (approx. 1000 µF per ampere motor current) parallel to the supply.
Operating modes	*Profile Position Mode*, *Profile Velocity Mode*, *Profile Torque Mode*, *Velocity Mode*, *Homing Mode*, *Interpolated Position Mode*, *Cyclic Sync Position Mode*, *Cyclic Sync Velocity Mode*, *Cyclic Synchronous Torque Mode*, *Clock-Direction Mode (not available for the IP65 variant)*

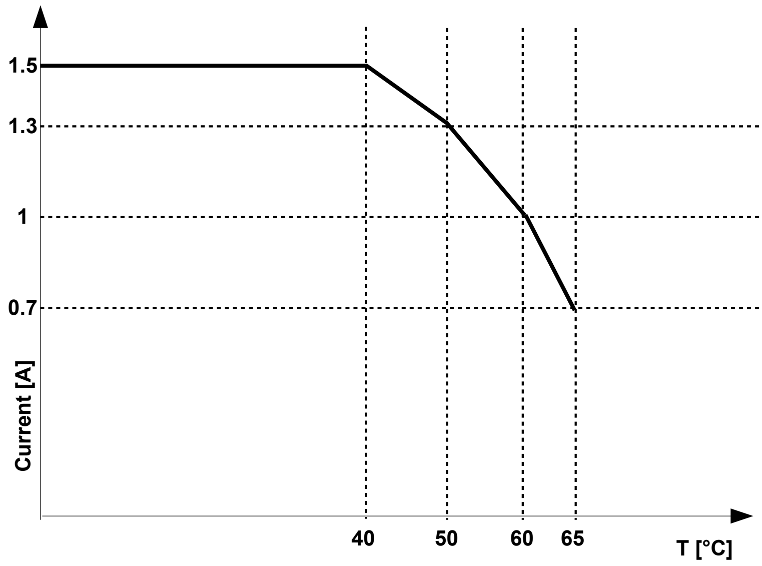
Property	Description / value
Set value setting / programming	*Clock-direction*, *analog*, *NanoJ program*
Interfaces	RS-485 (Modbus RTU)
Inputs	■ *PD1-C…-…-**OF**-…*: 3 digital (5/24 V switchable), 1 analog (12-bit resolution, 0 - 24 V, can also be read out as fourth digital input) ■ *PD1-C…-…-**20**-…*: 3 digital (5/24 V switchable), 1 analog (12-bit resolution, 0 - 24 V, can also be read out as fourth digital input) ■ *PD1-C…-…-**65**-…*: 1 digital (5/24 V switchable)
Outputs	■ *PD1-C…-…-**OF**-…*: 2 digital, *push-pull* (5/UB V switchable) ■ *PD1-C…-…-**20**-…*: 2 digital, *push-pull* (5/UB V switchable) ■ *PD1-C…-…-**65**-…*: 1 digital, *push-pull* (5/UB V switchable)
Protection circuit	Overvoltage and undervoltage protection Overtemperature protection (> 80° Celsius on the power board) Polarity reversal protection: In the event of a polarity reversal, a short-circuit will occur between supply voltage and GND over a power diode; a line protection device (fuse) is therefore necessary in the supply line. The values of the fuse are dependent on the application and must be dimensioned ■ greater than the maximum current consumption of the controller, ■ less than the maximum current of the voltage supply. If the fuse value is very close to the maximum current consumption of the controller, a medium / slow tripping characteristics should be used.

3.4 Overtemperature protection

Above a temperature of approx. 80 °C on the power board the power part of the controller switches off and the error bit is set (see objects 1001_h and 1003_h). After cooling down and confirming the error (see table for the controlword, "Fault reset"), the controller again functions normally.

Temperature-dependent power reduction

The following diagram shows the permissible continuous current as a function of the ambient temperature:



NOTICE

Aside from the motor, the exact temperature behavior is also dependent on the flange connection and the heat transfer there as well as on the convection in the application. For this reason, Nanotec recommends always performing an endurance test in the actual environment for applications in which current level and ambient temperature pose a problem.

Make sure that heat dissipation via the mounting surface and passive cooling or active ventilation are possible, so that the maximal ambient temperature stays within the limits.

3.5 Pin assignment

NOTICE

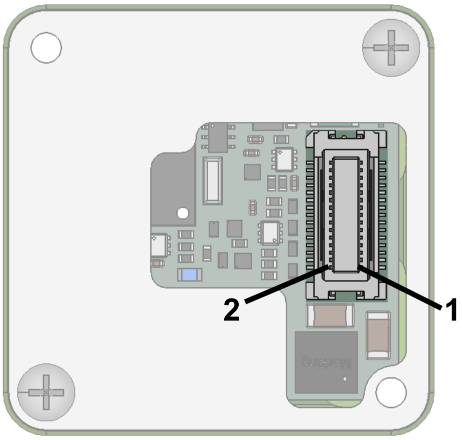
All pins with designation *GND* are internally connected.

3.5.1 Connections

PD1-...-OF-...

- Type: Molex 52991-0308

Pins 1 and 2 are marked in the following figure.

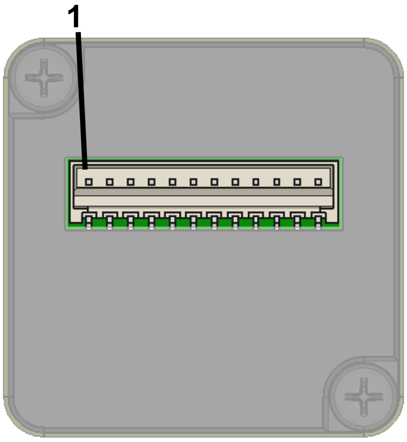


Pin(s)	Function	Note
2. 4, 6, 8, 10	GND	
1. 3, 5, 7, 9	+UB	12-30 V DC **Note:** You must connect all 5 pins to the supply voltage.
11	Digital input 1	5/24 V switchable with 323A$_h$, max. 1 MHz, clock input in clock-direction mode
12	Digital input 2	5/24 V switchable with 323A$_h$, max. 1 MHz, direction input in clock-direction mode
13	Digital input 3	5/24 V switchable with 323A$_h$
14	Analog input / digital input 4	12 bit, 0-30 V
15	Digital output 1	Push-pull, 5/+UB V switchable with 323A$_h$, max. 50 mA
16	*RS485+*	
17	Digital output 2	Push-pull, 5/+UB V switchable with 323A$_h$, max. 50 mA
18	*RS485-*	
19	reserved	do not connect
20	reserved	do not connect
21	reserved	do not connect
22	reserved	do not connect
23	reserved	do not connect
24	reserved	do not connect
25	reserved	do not connect
26	USER_SPI_NSS	*Chip Select* pin of the interface Generic SPI
27	USER_SPI_MISO	*MISO* pin of the interface Generic SPI
28	USER_SPI_SCK	*Clock* pin of the interface Generic SPI
29	USER_SPI_MOSI	*MOSI* pin of the interface Generic SPI
30	+3.3V	Output voltage, max. 100 mA

PD1-…-20-…

- Type: Molex 87437-1273

In the following figure, pin 1 is marked.

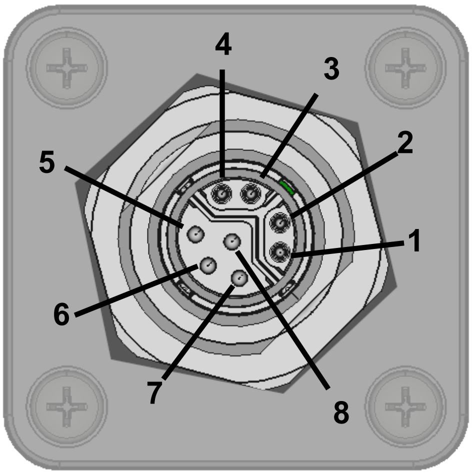


Pin	Function	Note
1	Digital output 1	Push-pull, 5/+UB V switchable with 323Ah, max. 50 mA
2	Digital output 2	Push-pull, 5/+UB V switchable with 323Ah, max. 50 mA
3	Digital input 1	5/24 V switchable with 323Ah, max. 1 MHz, clock input in clock-direction mode
4	Digital input 2	5/24 V switchable with 323Ah, max. 1 MHz, direction input in clock-direction mode
5	Digital input 3	5/24 V switchable with 323Ah
6	Analog input / digital input 4	12 bit, 0-30 V
7	*RS485+*	
8	*RS485-*	
9	reserved	do not connect
10	reserved	do not connect
11	+UB	12-30 V DC
12	GND	

PD1-…-65-…

- Type: M12, 8-pin, Y4-coded, male

The pin numbers are marked in the following figure.



Pin	Function	Note
1	*RS485+*	*RS485+* IN
2	*RS485-*	*RS485-* IN
3	*RS485+*	*RS485+* OUT
4	*RS485-*	*RS485-* OUT
5	Digital input 1	5/24 V switchable with $323A_h$
6	Digital output 1	Push-pull, 5/+UB V switchable with $323A_h$, max. 50 mA
7	+UB	12-30 V DC
8	GND	

Switching thresholds

The following switching thresholds apply for the digital inputs and the analog input (if available):

Max. voltage	Switching thresholds	
	On	Off
5 V	> 2 V	< 0,8 V
24 V	> 15 V	< 5 V

3.5.2 Voltage source

The operating or supply voltage supplies a battery, a transformer with rectification and filtering, or a switching power supply.

NOTICE

EMC: For a DC power supply line longer than 30 m or when using the motor on a DC bus, additional interference-suppression and protection measures are necessary.

► An EMI filter is to be inserted in the DC supply line as close as possible to the controller/ motor.

► Long data or supply lines are to be routed through ferrites.

3.5.3 Permissible operating voltage

The maximum operating voltage is 30 V. If the input voltage of the controller exceeds the threshold value set in 2034_h, the motor is switched off and an error triggered. The minimum operating voltage is 12 V DC. If the input voltage of the controller falls below the threshold value set in 2035_h, the motor is switched off and an error triggered.

A charging capacitor of at least 4700 µF / 50 V (approx. 1000 µF per ampere rated current) must be connected in parallel to the supply voltage to avoid exceeding the permissible operating voltage (e. g., during braking).

NOTICE

Damage to the controller and/or its power supply due to excitation voltage of the motor!

Voltage peaks during operation may damage the controller and possibly its power supply.

► Install suitable circuits (e.g., charging capacitor) that reduce voltage peaks.

► Use a power supply with protection circuit to protect against overvoltage.

3.5.4 RS-485 line polarization *and termination*

NOTICE

The controller is **not** equipped with line polarization and expects the master device to have one.

If the master device on the bus does not have line polarization of its own, a pair of resistors must be attached to the RS-485 balanced cables:

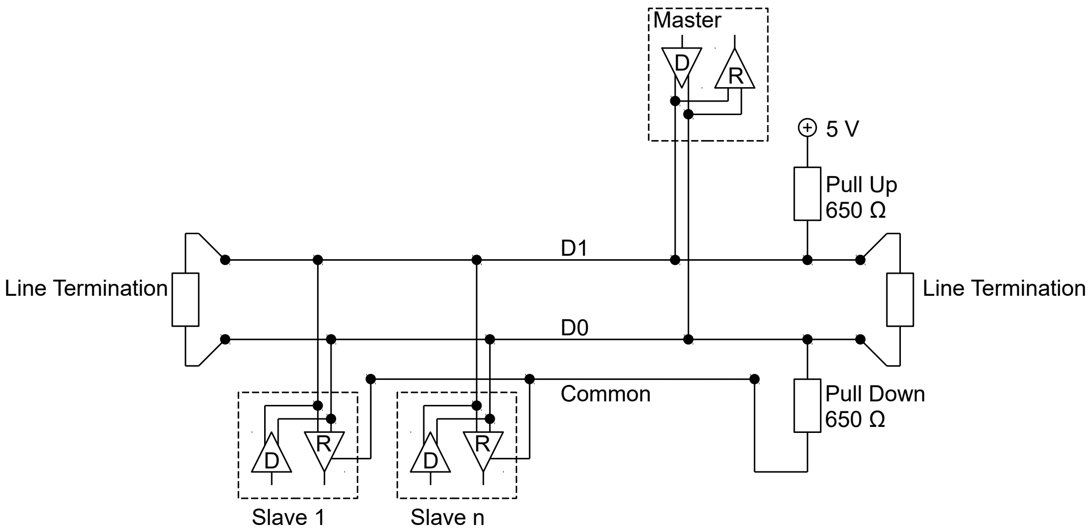
- A pull-up resistor to a 5V voltage on the RS-485+ (D1) cable
- A pull-down resistor to earth (GND) on the RS-485- (D0) cable

The value of these resistors must be between 450 ohm and 650 ohm. A 650 ohm resistor permits a higher number of devices on the bus.

In this case, a line polarization must be attached at a location for the entire serial bus. In general, this location should be on the master device or its connection. All other devices then no longer need to implement line polarization.

NOTICE

You must terminate the network with a termination resisotr of 150 Ohm on both line ends..


Master
D
R
5 V
Pull Up
650 Ω
D1
Line Termination
Line Termination
D0
Common
Pull Down
650 Ω
R
D
R
D
Slave 1
Slave n

4 Commissioning

Described in this chapter is how you establish communication with the controller and set the necessary parameters to make the motor ready for operation.

The controller also offers you the possibility to switch *special drive modes* on/off via object 4015_h. You can thereby control the motor directly via the inputs (analog input/clock-direction). See chapter Special drive modes (clock-direction and analog speed) for details.

Observe the following notes:

CAUTION!

Moving parts can cause hand injuries.

If you touch moving parts during running operation, hand injuries may result.

► Do not reach for moving parts during operation. After switching off, wait until all movements have ended.

CAUTION!

In free-standing operation, motor movements are uncontrolled and can cause injuries.

If the motor is unsecured, it can, e. g., fall down. Foot injuries or damage to the motor could occur.

► If you operate the motor free-standing, observe the motor, switch it off immediately in the event of danger and make certain that the motor cannot fall down.

CAUTION!

Moving parts can catch hair and loose clothing.

During running operation, moving parts can catch hair or loose clothing, which may lead to injuries.

► If you have long hair, wear a hairnet or take other suitable protective measures when near moving parts. Do not work with loose clothing or ties near moving parts.

CAUTION!

Risk of overheating or fire if there is insufficient cooling!

If cooling is insufficient or if the ambient temperature is too high, there is a risk of overheating or fire.

► During use, make certain that the cooling and environmental conditions are ensured.

NOTICE

EMC: Current-carrying cables – particularly around supply cables – produce electromagnetic alternating fields. These can interfere with the motor and other devices.

Suitable measures may be:

- ▶ Use shielded cables and earth the cable shielding on both ends over a short distance.
- ▶ Keep power supply cables as short as possible.
- ▶ Use cables with cores in twisted pairs.
- ▶ Earth motor housing with large contact area over a short distance.
- ▶ Lay supply and control cables separately.

4.1 Configuring via Modbus RTU

Described in the following chapters is how you can establish the communication.

The controller is set to slave address ex works , baud rate 19200 baud, even parity, 1 stop bit. All changes take effect only after the controller is restarted.

4.1.1 Communication settings

Configuration	Object	Value range	Factory settings
Slave address	2028_h	1 to 247	5
Baud rate	$202A_h$	7200 to 256000	19200
Parity	$202D_h$	■ None: 0x00 ■ Even: 0x04 ■ Odd: 0x06	0x04 (Even)

The number of data bits is always "8" here. The number of stop bits is dependent on the parity setting:

- No parity: 2 stop bits
- "Even" or "Odd" parity: 1 stop bit

The following baud rates are supported:

- 7200
- 9600
- 14400
- 19200
- 38400
- 56000
- 57600
- 115200
- 128000
- 256000

You must save the changes by writing value "65766173_h" in object 1010_h:$0B_h$. The changes are not taken over until after the controller has been restarted.

4.2 Auto setup

To determine a number of parameters related to the motor and the connected sensors (encoders/Hall sensors), you must perform an auto setup.

TIP
As long as the motor connected to the controller or the sensors for feedback (encoders/Hall sensors) are not changed, auto setup is only to be performed once during initial commissioning.

NOTICE
Note the following prerequisites for performing the auto setup: ▶ The motor must be load-free. ▶ The motor must not be touched. ▶ The motor must be able to turn freely in any direction. ▶ No NanoJ programs may be running (object 2300_h:00_h bit 0 = "0").

TIP
Execution of the auto setup requires a relatively large amount of processor computing power. During the auto setup, this may result in fieldbuses not being operated in a timely manner.

4.2.1 Parameter determination

Auto setup determines various parameters of the connected motor and of the present sensors by means of multiple test runs and measurement runs. To a certain extent, the type and number of parameters are dependent on the respective motor configuration.

Parameter	All motors independent of the configuration
Motor type (stepper motor or BLDC motor)	✓
Winding resistance	✓
Winding inductance	✓
Interlinking flux	✓

NOTICE
It is not possible to determine the interlinking flux on motors whose windings have widely differing inductances. These motors are, therefore, not suitable for sensorless *closed-loop* operation.

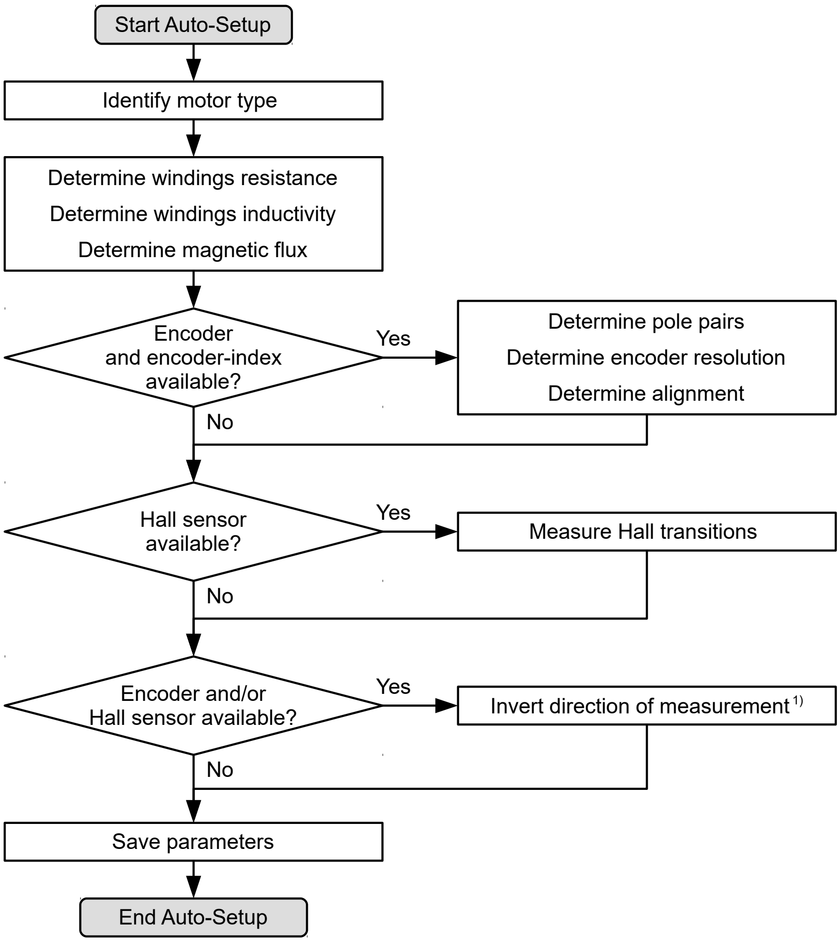
Parameter	Motor without encoder	Motor with encoder and index	Motor with encoder without index
Encoder resolution	-	✓	---
Alignment (shifting of the electrical zero to the index)	-	✓	---

Parameter	Motor without Hall sensor	Motor with Hall sensor
Hall transitions	-	✓

4.2.2 Execution

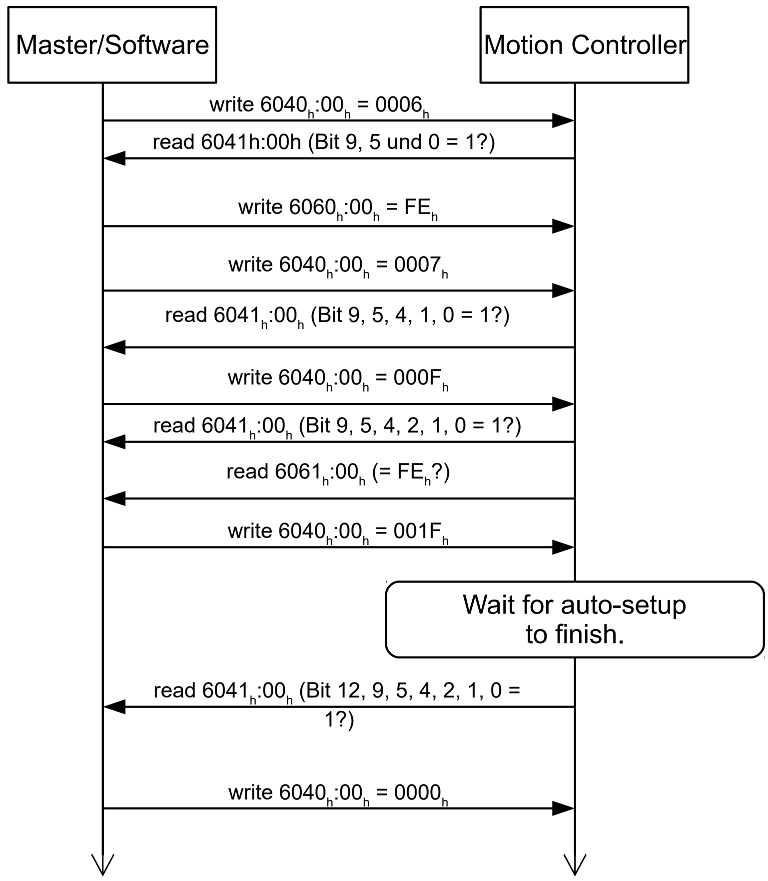
1. To preselect the *auto setup* operating mode, enter the value "-2" (="FE_h") in object $6060_h:00_h$.
   The *power state machine* must now switch to the *Operation enabled* state, see CiA 402 Power State Machine.
2. Start *auto setup* by setting bit 4 *OMS* in object $6040_h:00_h$ (controlword).

   While the auto setup is running, the following tests and measurements are performed in succession:



1) To determine the values, the direction of the measurement method is reversed and edge detection re-evaluated.

Value 1 in bit 12 *OMS* in object $6041_h:00_h$ (statusword) indicates that the auto setup was completely executed and ended. In addition, bit 10 *TARG* in object $6041_h:00_h$ can be used to query whether (= "1") or not (= "0") an encoder index was found.



4.2.3 Parameter memory

After a successful *auto setup*, the determined parameter values are automatically taken over into the corresponding objects and stored with the storage mechanism, see Saving objects and 1010h Store Parameters. Categories *Drive* 1010_h:05_h and *Tuning* 1010_h:06_h are used.

CAUTION!

Uncontrolled motor movements!

After the auto setup, the internal coordinate system is no longer valid. Unforeseen reactions can result.

► Restart the device after an auto setup. Homing alone does not suffice.

4.3 Special drive modes (clock-direction and analog speed)

NOTICE

These modes are not available for the IP65 variant.

You have the possibility to control the motor directly via the clock and direction input or the analog input by activating the *special drive modes*.These include:

- Clock-direction
- Analog speed
- Test run with 30 rpm

You can also determine the control mode – *open-loop* or *closed-loop*.

Digital input 3 serves here as an enable (see Connections).

NOTICE

After activating the *special drive modes*, the state of the CiA 402 Power State Machine is controlled only via a digital input (enable). State changes that are requested in object 6040_h (controlword) have no effect.

4.3.1 Activation

To activate the *special drive modes*, you must enter the value "2" in 4015_h:01_h. In 4015_h:02_h, set the mode by writing a value between "00"$_h$ and "0F"$_h$.

The following table lists all possible modes and their value for 4015:02_h:

Value	Mode			
00_h/01_h	Clock-direction	-	-	Open-Loop
02_h	Clock-direction (test run)	Test run with 30 rpm	Clockwise direction of rotation	Open-Loop
03_h	Clock-direction (test run)	Test run with 30 rpm	Counterclockwise direction of rotation	Open-Loop
04_h	Analog speed	Direction via "Direction" input	Maximum speed 1000 rpm	Open-Loop
05_h	Analog speed	Direction via "Direction" input	Maximum speed 100 rpm	Open-Loop
06_h	Analog speed	Offset 5 V (joystick mode)	Maximum speed 1000 rpm	Open-Loop
07_h	Analog speed	Offset 5 V (joystick mode)	Maximum speed 100 rpm	Open-Loop
08_h/09_h	Clock-direction	-	-	Closed-Loop
$0A_h$	Clock-direction (test run)	Test run with 30 rpm	Clockwise direction of rotation	Closed-Loop
$0B_h$	Clock-direction (test run)	Test run with 30 rpm	Counterclockwise direction of rotation	Closed-Loop
$0C_h$	Analog speed	Direction via "Direction" input	Maximum speed 1000 rpm	Closed-Loop
$0D_h$	Analog speed	Direction via "Direction" input	Maximum speed 100 rpm	Closed-Loop
$0E_h$	Analog speed	Offset 5 V (joystick mode)	Maximum speed 1000 rpm	Closed-Loop
$0F_h$	Analog speed	Offset 5 V (joystick mode)	Maximum speed 100 rpm	Closed-Loop

You must save object 4015_h (*application category*) (see chapterSaving objects); the changes do not take effect until after the controller is restarted.

4.3.2 Clock-direction

The controller internally sets the operating mode to clock-direction. You must connect the *enable*, *clock* and *direction* inputs (see chapter Connections).

4.3.3 Analog speed

The controller internally sets the operating mode to Velocity. To preset the speed, the voltage on the analog input is used and the corresponding target speed is written in 6042_h.

4.3.3.1 Maximum speed

The maximum speed can be changed between 100 rpm and 1000 rpm; the controller automatically adapts the scaling in 604_h here.

NOTICE

If you would like to change to a different mode afterwards, you must adapt or reset the scaling in $604C_h$ if necessary.

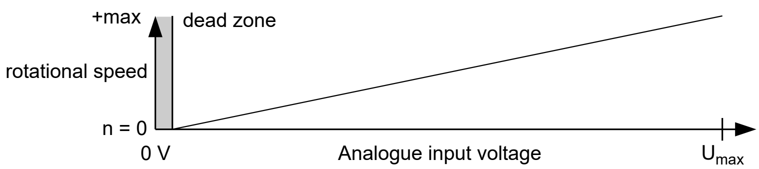
If a different speed is necessary, it can be set using the scaling factor for the speed (object $604C_h$) or the analog value (see Analog inputs).

4.3.3.2 Computation of the analog voltage

There are two modes for calculating the analog input voltage.

Normal mode

You must connect the *enable*, *direction* and *analog inputs* (see chapter Connections). The maximum analog voltage corresponds to the maximum speed. The direction is preset here via the direction input. If there is no signal at the direction input, the motor turns clockwise (when looking at the drive shaft). There is a dead zone from 0 V to 20 mV in which the motor does not move.



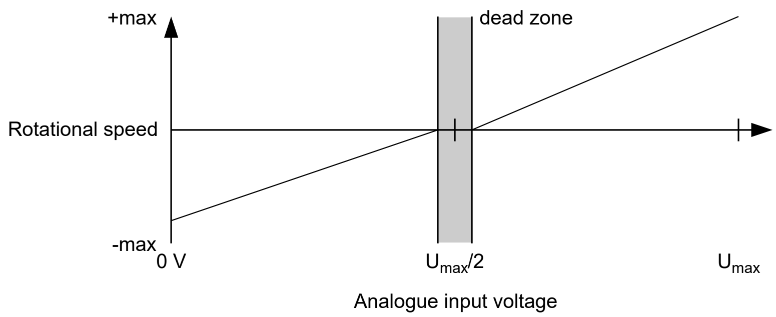
Joystick mode

You must connect the *release input* and the *analog input* (see chapter Connections). The half of the maximum analog voltage corresponds to the speed 0; the controller automatically adapts the offset in 3321_h here.

NOTICE

If you would like to change to a different mode afterwards, you must adapt or reset the offset in 3321_h if necessary.

If the voltage drops below half, the speed increases in the negative direction. If the speed rises above half, the speed increases likewise in the positive direction. The dead zone here extends from $U_{max}/2 \pm$ 20 mV.



4.3.4 Test run with 30 rpm

The motor rotates at 30 rpm if the *enable input* is set.

5 General concepts

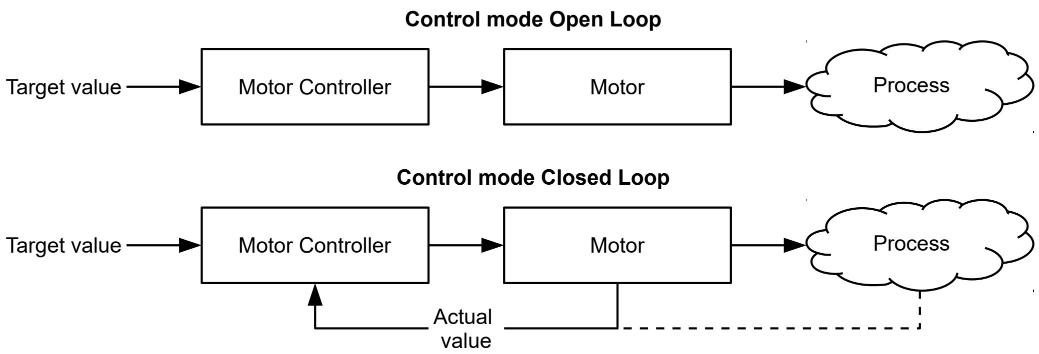
5.1 Control modes

5.1.1 General

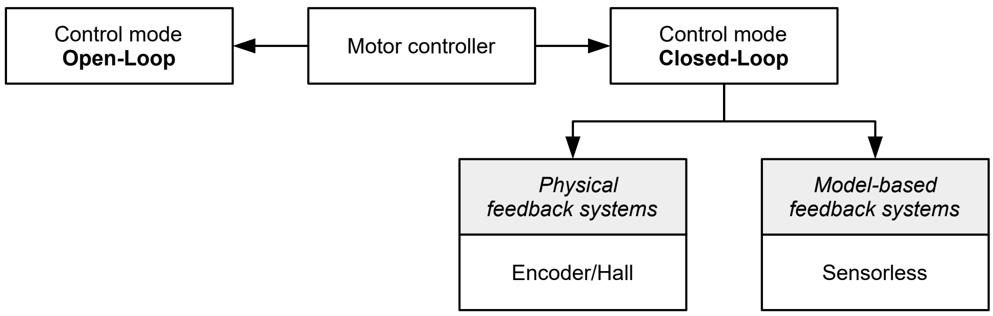
The control mode of systems without feedback is called *open-loop*, the mode with feedback is called *closed-loop*. In the *closed-loop* control mode, it is initially irrelevant whether the fed back signals come from the motor itself or from the influenced process.

For controllers with feedback, the measured control variable (actual value) is constantly compared with a set point (set value). In the event of deviations between these values, the controller readjusts according to the specified control parameters.

Pure controllers, on the other hand, have no feedback for the value that is to be regulated. The set point (set value) is only specified.



In addition to the physical feedback systems (e.g., via encoders or Hall sensors), model-based feedback systems, collectively referred to as *sensorless* systems, are also used. Both feedback systems can also be used in combination to further improve the control quality.



Summarized in the following are all possible combinations of control modes and feedback systems with respect to the motor technology. Support of the respective control mode and feedback is controller-specific and is described in chapters *Pin assignment* and operating modes.

Control mode	Stepper motor	BLDC motor
Open-Loop	yes	no
Closed-Loop	yes	yes

Feedback	Stepper motor	BLDC motor
Hall	no	yes
Encoder	yes	yes

Feedback	Stepper motor	BLDC motor
Sensorless	yes	yes

Various operating modes can be used depending on the control mode. The following list contains all the types of operation that are possible in the various control modes.

1) The Profile Torque and Cyclic Synchronous Torque torque operating modes are not possible in the *open-loop* control mode due to a lack of feedback.

2) Exception: Homing on block is not possible due to a lack of feedback.

3) Because ramps and speeds in operating modes Cyclic Synchronous Position and Cyclic Synchronous Velocity follow from the specified points of the master, it is not normally possible to preselect these parameters and to ascertain whether a step loss can be excluded. It is therefore not advisable to use these operating modes in combination with *open-loop* control mode.

5.1.2 Open-Loop

5.1.2.1 Introduction

Open-loop mode is only used with stepper motors and is, by definition, a control mode without feedback. The field rotation in the stator is specified by the controller. The rotor directly follows the magnetic field rotation without step losses as long as no limit parameters, such as the maximum possible torque, are exceeded. Compared to *closed-loop*, no complex internal control processes are needed in the controller. As a result, the requirements on the controller hardware and the controller logic are very low. *Open-loop* mode is used primarily with price-sensitive applications and simple movement tasks.

Because, unlike *closed-loop*, there is no feedback for the current rotor position, no conclusion can be drawn on the counter torque being applied to the output side of the motor shaft. To compensate for any torque fluctuations that arise on the output shaft of the motor, in *open-loop* mode, the controller always supplies the maximum possible (e.g., specified by parameters) set current to the stator windings over the entire speed range. The high magnetic field strength thereby produced forces the rotor to assume the new steady state in a very short time. This torque is, however, opposite that of the inertia of the rotor and overall system. Under certain operating conditions, this combination is prone to resonances, comparable to a spring-mass system.

5.1.2.2 Commissioning

To use *open-loop* mode, the following settings are necessary:

- In object 2030_h (Pole Pair Count), enter the number of pole pairs (see motor data sheet: for a stepper motor with 2 phases, a step angle of 1.8° corresponds to 50 pole pairs and 0.9° corresponds to 100 pole pairs).
- In object 2031_h:00_h, enter the maximum permissible motor current (motor protection) in mA (see motor data sheet)
- In object 6075_h:00_h, enter the rated current of the motor in mA (see motor data sheet).
- In object 6073_h:00_h, enter the maximum current (for a stepper motor, generally corresponds to the rated current, bipolar) in tenths of a percent of the set rated current (see motor data sheet). Factory settings: "1000", which corresponds to 100% of the value in 6073_h. A value greater than "1000" is limited internally to "1000".
- In object 3202_h (Motor Drive Submode Select), set bit 0 (CL/OL) to the value "0".

Nanotec recommends the current reduction on motor standstill in order to reduce the power loss and heat build-up:

- In object 2036_h (open-loop current reduction idle time), the time in milliseconds is specified that the motor must be at a standstill (set value is checked) before current reduction is activated.
- In object 2037_h (open-loop current reduction value/factor), the root mean square is specified to which the rated current is to be reduced if current reduction is activated in *open loop* and the motor is at a standstill.

5.1.2.3 Optimizations

Depending on the system, resonances may occur in *open-loop* mode; susceptibility to resonances is particularly high at low loads. Practical experience has shown that, depending on the application, various measures are effective for largely reducing resonances:

- Reduce or increase current, see objects 6073_h and 6075_h, respectively. An excessive torque reserve promotes resonances.
- Reduce or increase the operating voltage, taking into account the product-specific ranges (with sufficient torque reserve). The permissible operating voltage range can be found in the product data sheet.
- Optimize the control parameters of the current controller via objects 3210_h:09_h (I_P) and 3210_h:$0A_h$ (I_I) (generally not necessary).
- Adjustments to the acceleration, deceleration and/or target speed depending on the selected control mode:

  Profile Position operating mode

  Objects 6083_h (Profile Acceleration), 6084_h (Profile Deceleration) and 6081_h (Profile Velocity).

  Velocity operating mode

  Objects 6048_h (Velocity Acceleration), 6049_h (Velocity Deceleration) and 6042_h (Target Velocity).

  Profile Velocity operating mode

  Objects 6083_h (Profile Acceleration), 6084_h (Profile Deceleration) and 6081_h (Profile Velocity).

  Homing operating mode

  Objects $609A_h$ (Homing Acceleration), 6099_h:01_h (Speed During Search For Switch) and 6099_h:02_h (Speed During Search For Zero).

  Interpolated Position Mode operating mode

  The acceleration and deceleration ramps can be influenced with the higher-level controller.

  Cyclic Synchronous Position operating mode

  The acceleration and deceleration ramps can be influenced via the external "position specification / time unit" targets.

  Cyclic Synchronous Velocity operating mode

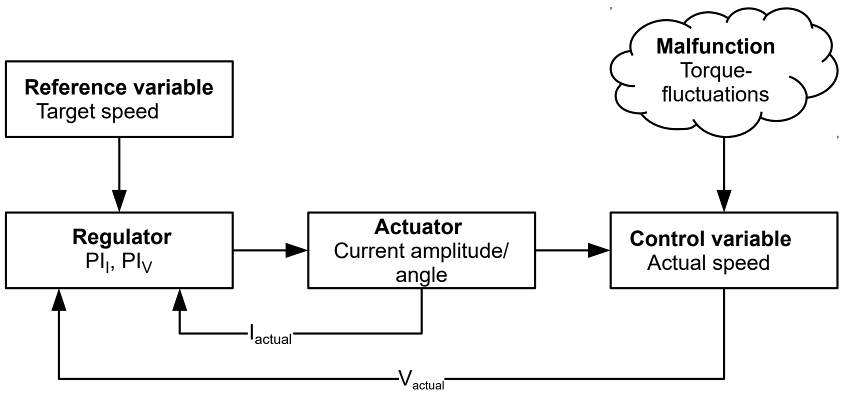
  The acceleration and deceleration ramps can be influenced via the external "position specification / time unit" targets.

5.1.3 Closed-Loop

5.1.3.1 Introduction

The *closed-loop* theory is based on the idea of a control loop. A disturbance acting on a system should be compensated for quickly and without lasting deviation to adjust the control variable back to the set point.

Closed loop using a speed control as an example:



PI_I	=	Proportional-integral current control loop
PI_V	=	Proportional-integral velocity control loop
I_{actual}	=	Actual current
V_{actual}	=	Actual speed

The *closed-loop* method is also referred to as "sine commutation via an encoder with field-oriented control". At the heart of *closed-loop* technology is the performance-adjusted current control as well as the feedback of the actual values of the process. Using sensor signals, the rotor orientation is recorded and sinusoidal phase currents generated in the motor windings. Vector control of the magnetic field ensures that the magnetic field of the stator is always perpendicular to that of the rotor and that the field strength corresponds precisely to the desired torque. The current thereby controlled in the windings provides a uniform motor force and results in an especially smooth-running motor that can be precisely regulated.

The feedback of the control variables necessary for *closed-loop* mode can be realized with various technologies. In addition to the physical feedback with encoders or Hall sensors, it is also possible to virtually record the motor parameters through a software-based model calculation. Physical variables, such as speed or back-EMF, can be reconstructed with the help of a so-called "observer" from the data of the current controller. With this sensorless technology, one has a "virtual rotary encoder", which – above a certain minimum speed – supplies the position and speed information with the same precision as a real optical or magnetic encoder.

All controllers from Nanotec that support *closed-loop* mode implement a field oriented control with sine commutated current control. Thus, the stepper motors and BLDC motor are controlled in the same way as a servo motor. With *closed-loop* mode, step angle errors can be compensated for during travel and load angle errors corrected within one full step.

5.1.3.2 Controller structure

The controller consists of three cascaded PI controllers (proportional-integral): the current controller (commutation), the velocity controller and the position controller.

The current controller is active in all operating modes. The velocity controller is as well with the sole exception of the "Real Torque" modes (torque mode without speed limiting if bit 5 in 3202_h is set to "1").

The position controller is active in the following operating modes:

- Profile Position
- Homing
- Interpolated Position Mode
- Cyclic Synchronous Position
- Velocity/Profile Velocity/Cyclic Synchronous Velocity if bit 1 in 3202_h is set to "1"

Each controller consists of a proportional component with the *gain factor* K_p and an integral component with the *integrator time* T_i. The control variable (the output signal of the controller, which is the set point for the next controller) is limited by the maximum speed (position controller), the maximum current (velocity controller) or the maximum PWM signal (current controller), respectively.

Object	Name	Unit	Description
$321A_h$:01_h	Current controller Proportional Gain Kp for Iq	[mV/A]	Proportional component of torque-forming component
$321A_h$:02_h	Current controller Integrator Time Ti for Iq	[µs]	Integrator time of torque-forming component
$321A_h$:03_h	Current controller Proportional Gain Kp for Id	[mV/A]	Proportional component of field-forming component
$321A_h$:04_h	Current controller Integrator Time Ti for Id	[µs]	Integrator time of field-forming component
$321B_h$:01_h	Velocity controller Proportional Gain Kp	[mA/Hz]	Proportional component
$321B_h$:02_h	Velocity controller Integrator Time Ti	[µs]	Integrator time
$321C_h$:01_h	Position controller Proportional Gain Kp	[Hz] (Controller deviation in mech. revolutions per second)	Proportional component
$321C_h$:02_h	Position controller Integrator Time Ti	[µs]	Integrator time

The *gain factor* K_p has a direct influence on the current control variable: at the same deviation, the control variable is proportional to the gain factor.

Each controller also has an integral component that is determined by the *integrator time* (T_i). The smaller the integrator time, the faster the control variable increases. If the integrator time is 0, the integral component is internally set to "0" and the controller only has the proportional component.

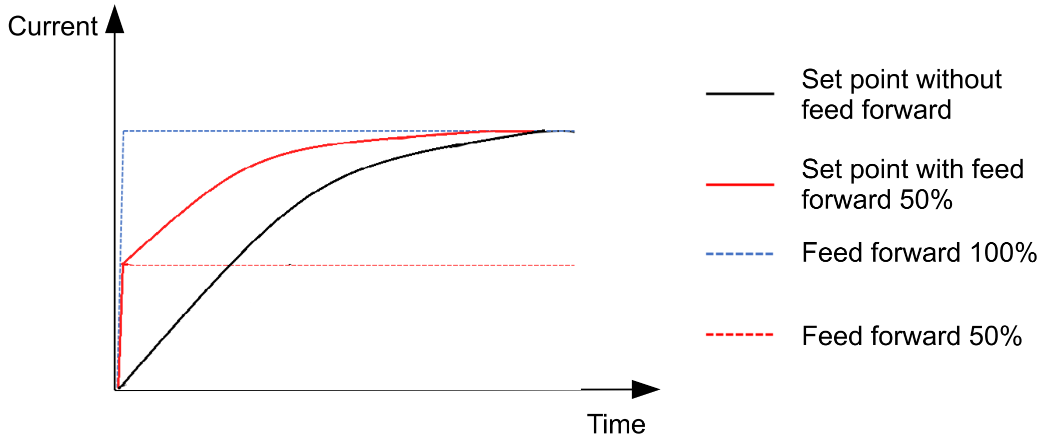
5.1.3.3 Feed forward

It is also possible to set a *velocity feed forward*, an *acceleration feed forward* (that corresponds to a torque/current value) and a *voltage feed forward*.

You can use the *feed forward* to add an already known or anticipated control variable to the set point ("predictive"). You can, e. g., compensate for the inertia of the load by adding an acceleration feed forward value to the output of the velocity controller.

The feed forward values are additionally fed to the speed/current control loop or added to the voltage value and are immediately available. A more dynamic control can thereby be achieved.

The following figure shows the current (produced by the acceleration) during the acceleration phase as a function of the *acceleration feed forward*. At a feed forward value of "50%", the current is at "50%" already at the start of the acceleration phase; the current controller is thereby "relieved".

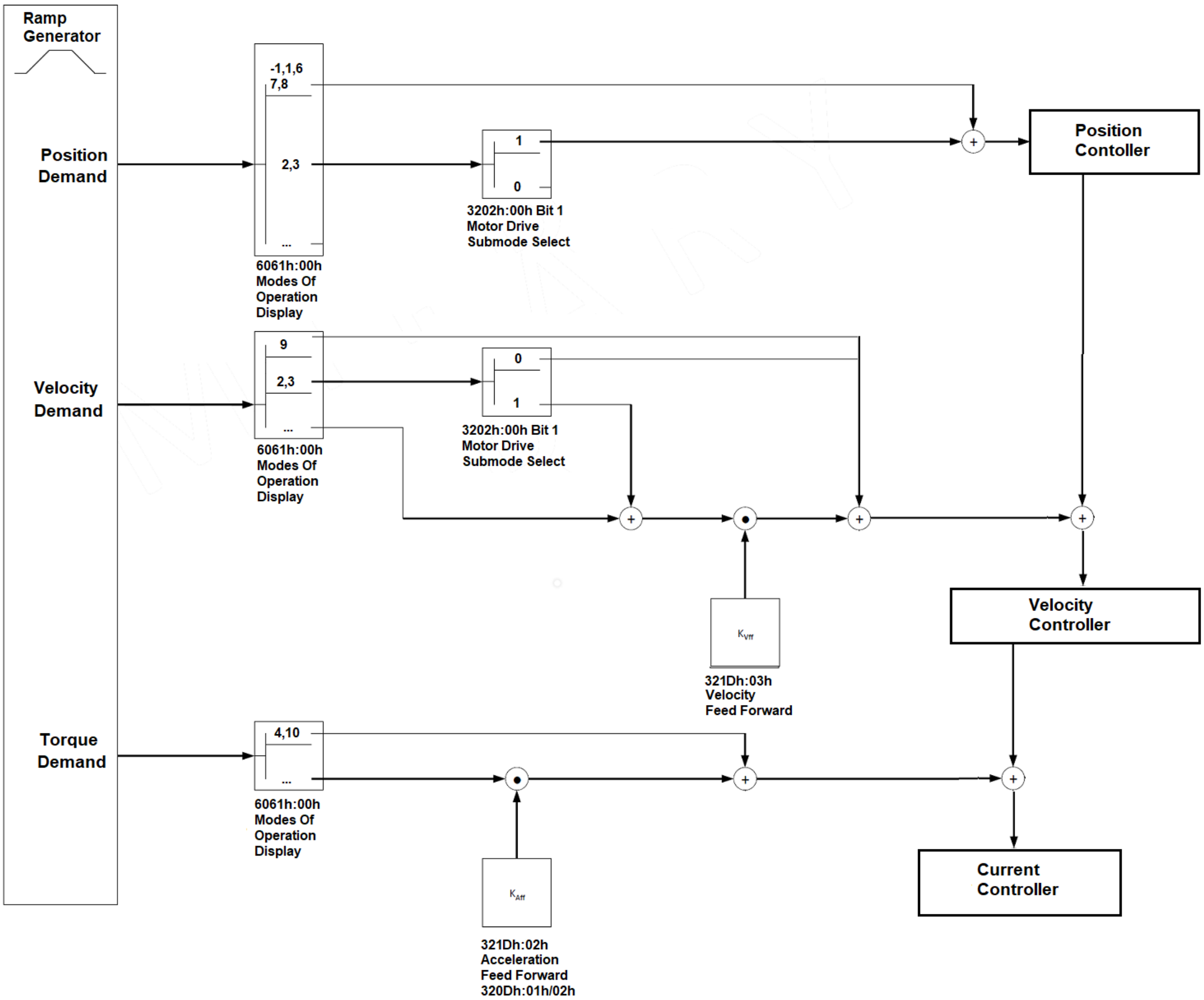


The factor for the *velocity feed forward* is set in object 321D$_h$:03$_h$ in tenths of a percent of the output of the ramp generator (606B$_h$) and added to the output of the position controller before the velocity controller. The *velocity feed forward* is active in all modes with position control loop:

- Profile Position
- Homing
- Interpolated Position Mode
- Cyclic Synchronous Position
- Velocity/Profile Velocity if bit 1 in 3202$_h$ is set to "1"

The factor for the *acceleration feed forward* is set in object 321D$_h$:02$_h$ in tenths of a percent of the factor of 320D$_h$ and multiplied by the output of the ramp generator (6074$_h$). The value is added to the output of the velocity controller before the current controller. The *acceleration feed forward* is active in all modes, with the exception of the torque modes.

The following figure shows the cases in which the feed forward is active and the position of the feed forward within the controller cascade.



The d- and q-current controllers have a reciprocal influence on one another. To neutralize this coupling, voltages are precalculated by the controller and added to the voltages calculated by the current controller. You can adjust this decoupling with object $321D_h:01_h$ (factory setting 1000 ‰).

The voltage required for a desired current can be precalculated based on the value for the ohmic resistance determined in auto setup. With object $321D_h:04_h$, you can adjust the precalculated voltage (factory setting 0 ‰). If this voltage is immediately available, the actual current can very quickly follow the set value and support the integral component of the current controller. When using this *voltage feed forward*, you should increase the Ti time values of the current controller in object $321A_h$ accordingly (significantly).

5.1.3.4 Commissioning

An auto setup should be performed before using *closed-loop* mode. The auto setup operating mode automatically determines the necessary parameters (e.g., motor data, feedback systems) that are necessary for optimum operation of the field oriented control. All information necessary for performing the auto setup can be found in chapter Auto setup.

Bit 0 in 3202_h must be set . The bit is set automatically after a successfully completed auto setup.

5.1.3.5 Optimizations

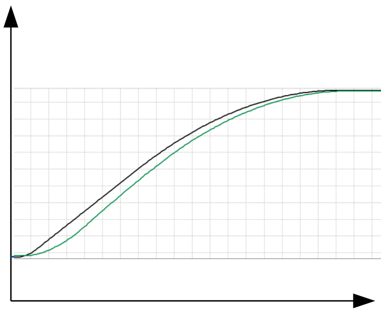
In *closed-loop*, the measured control variable (actual value) is constantly compared with a set point (set value). In the event of deviations between these values, the controller readjusts according to the specified control parameters.

The objective of control parameter optimization (the so-called *tuning* of the controller) is the smoothest possible running of the motor, high accuracy and high dynamics in the reaction of the controller to faults. All control deviations should be eliminated as quickly as possible.

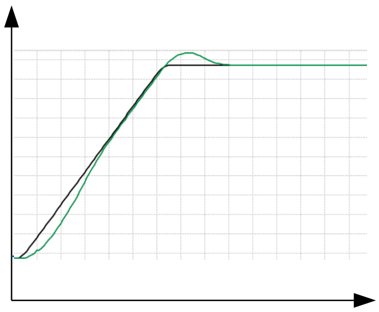
Due to the cascaded Controller structure, it is useful to start the optimization of the inner-most controller (current controller) before the velocity and – if applicable – the position controller are optimized. Each of the three controllers consists of a proportional and an integral component, which should normally be adjusted in this order.

The following figures show the reaction of the controller to a change in set value.

If the proportional component is too small, the actual value remains below the set value. A proportional component that is too large, on the other hand, results in "overshooting".

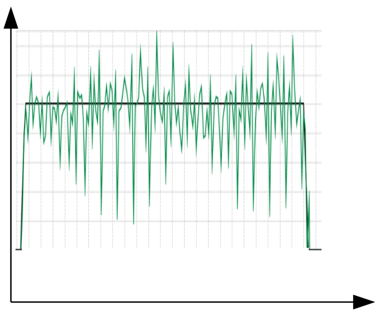


P-part too small

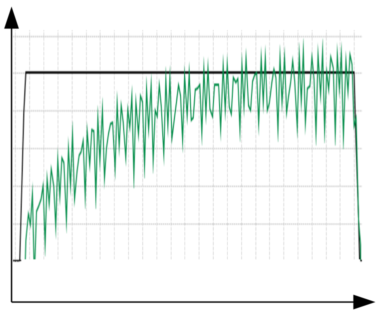


P-part too big

If the integrator time is too small, the system tends toward oscillations. If the integrator time is too large, the deviations are compensated for too slowly.



Ti too small



Ti too big

CAUTION!

Risk of injury through uncontrolled motor movements!

Incorrect control parameters may result in an unstable control behavior. Unforeseen reactions can result.

► Increase the control parameters slowly and incrementally. Do not increase these further if you notice strong vibrations/oscillations.

► Do not reach for moving parts during operation. After switching off, wait until all movements have ended.

5.2 CiA 402 Power State Machine

5.2.1 State machine

5.2.1.1 CiA 402

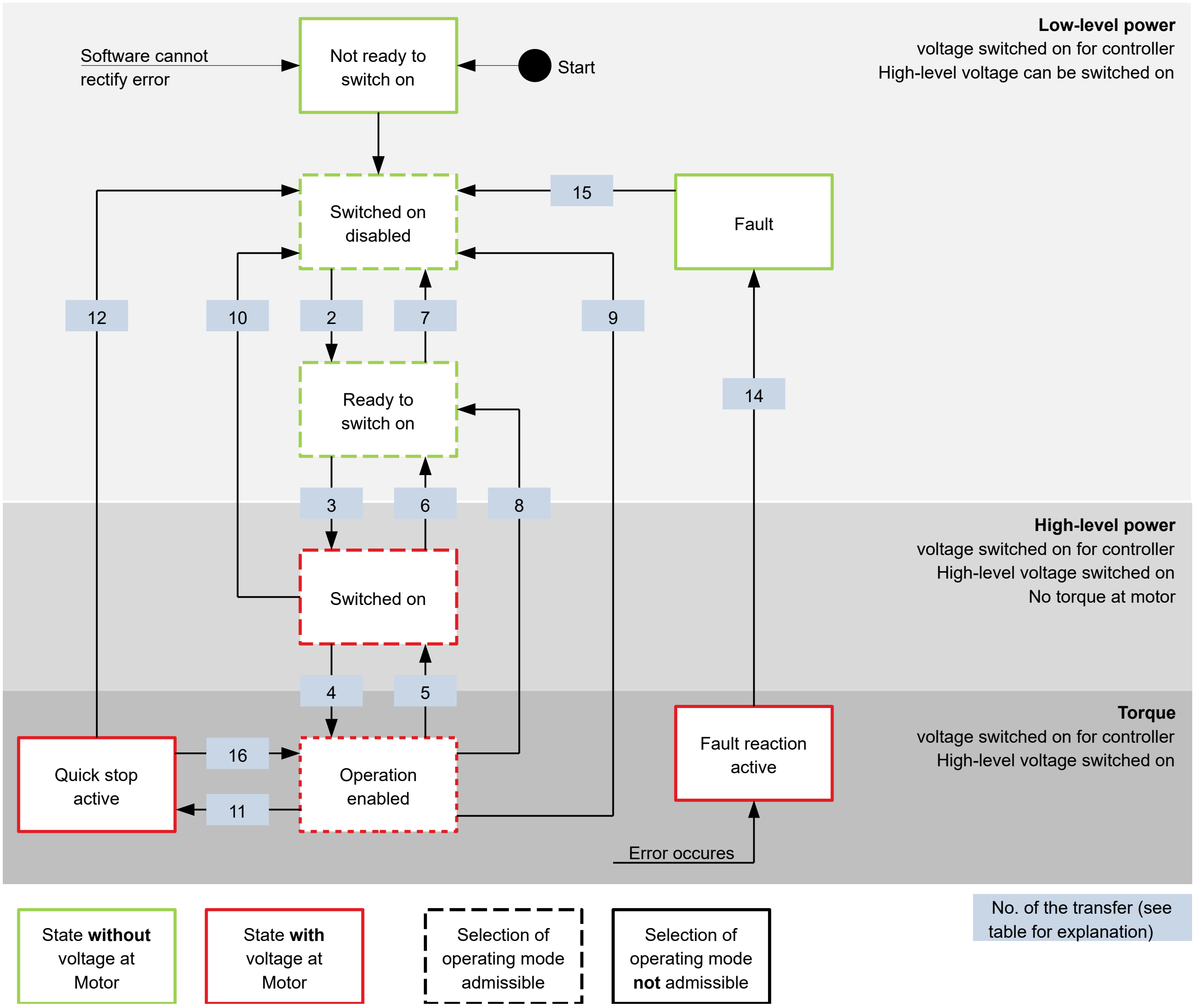
To switch the controller to the ready state, it is necessary to run through a *state machine*. This is defined in *CANopen standard 402*. State changes are requested in object 6040$_h$ (controlword). The actual state of the state machine can be found in object 6041$_h$ (statusword).

5.2.1.2 Controlword

State changes are requested via object 6040$_h$ (controlword).

State transitions

The diagram shows the possible state transitions.



Listed in the following table are the bit combinations for the controlword that result in the corresponding state transitions. An X here corresponds to a bit state that requires no further consideration. Exceptions are the resetting of the error (fault reset) and the changeover from *Quick Stop Active* to *Operation Enabled*: the transition is only requested by the rising edge of the bit.

Command	Bit in object 6040_h					Transition
	Bit 7	Bit 3	Bit 2	Bit 1	Bit 0	
Shutdown	0	X	1	1	0	2, 6, 8
Switch on	0	0	1	1	1	3
Disable voltage	0	X	X	0	X	7, 10, 9, 12
Quick stop	0	X	0	1	X	11
Disable operation	0	0	1	1	1	5
Enable operation	0	1	1	1	1	4
Enable operation after Quick stop	0	1		1	1	16
Fault / warning reset		X	X	X	X	15

5.2.1.3 Statusword

Listed in the following table are the bit masks that break down the state of the controller.

Statusword (6041_h)	State
xxxx xxxx x0xx 0000	Not ready to switch on
xxxx xxxx x1xx 0000	Switch on disabled
xxxx xxxx x01x 0001	Ready to switch on
xxxx xxxx x01x 0011	Switched on
xxxx xxxx x01x 0111	Operation enabled
xxxx xxxx x00x 0111	Quick stop active
xxxx xxxx x0xx 1111	Fault reaction active
xxxx xxxx x0xx 1000	Fault

After switching on and successfully completing the self-test, the controller reaches the *Switch on disabled* state.

5.2.1.4 Operating mode

The operating mode is set in object 6060_h. The actually active operating mode is displayed in 6061_h.

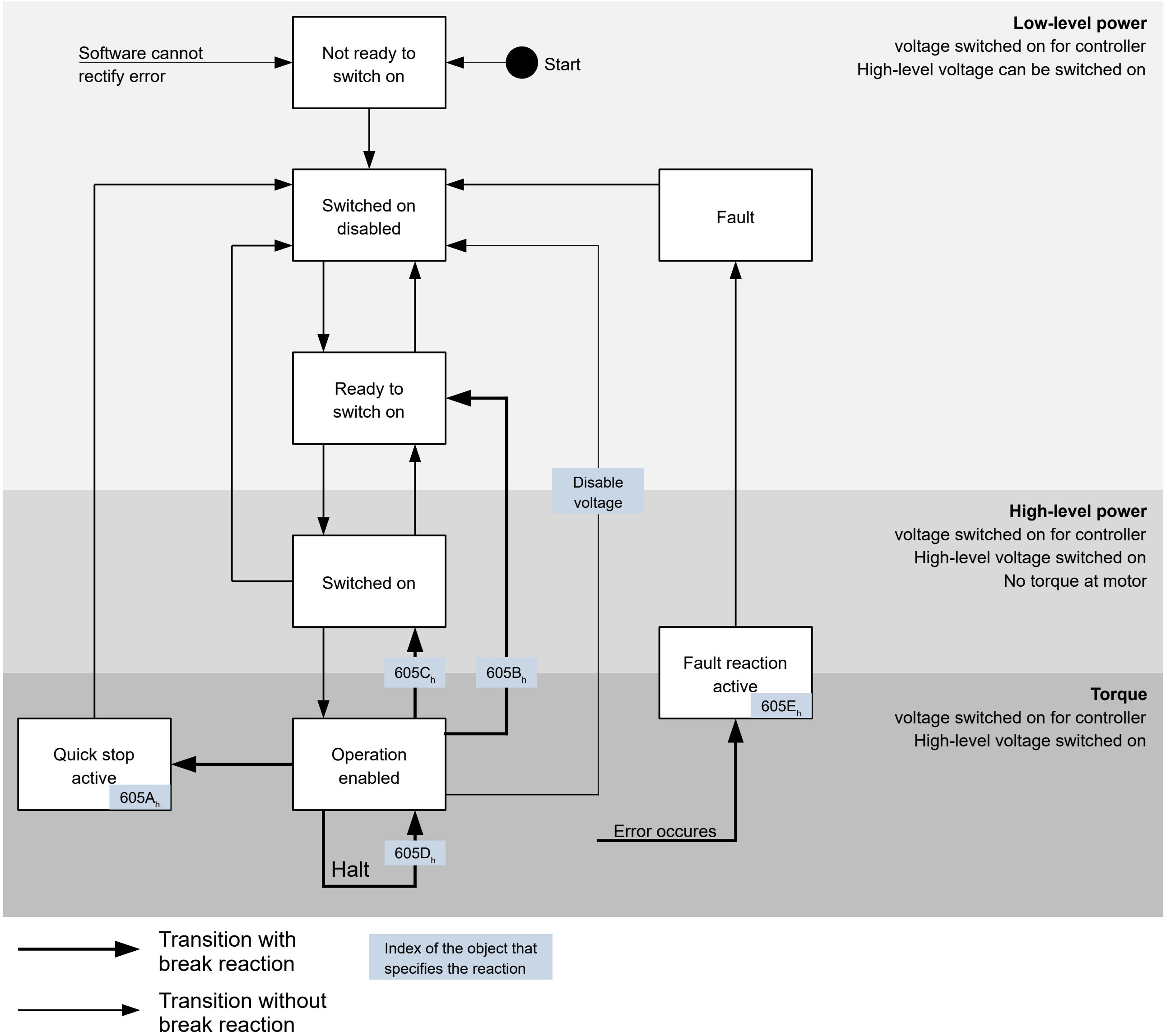
The operating mode can be set or changed at any time.

5.2.2 Behavior upon exiting the *Operation enabled* state

5.2.2.1 Halt motion reactions

Various halt motion reactions can be programmed upon exiting the *Operation enabled* state.

The following graphic shows an overview of the halt motion reactions.



5.2.2.2 Quick stop active

Transition to the *Quick stop active* state (quick stop option):

In this case, the action stored in object 605A$_h$ is executed (see following table).

Value in object 605A$_h$	Description
0	Switch off driver without deceleration ramp; drive function blocked – motor can turn freely
1	Braking with *slow down ramp* (deceleration ramp depending on operating mode) and subsequent state change to *Switch on disabled*
2	Braking with *quick stop ramp* (6085$_h$) and subsequent state change to *Switch on disabled*
5	Braking with *slow down ramp* (deceleration ramp depending on operating mode) and subsequent state change to *Quick stop active*; control does not switch off and the motor remains energized. You can switch back to the *Operation enabled* state.

Value in object $605A_h$	Description
6	Braking with *quick stop ramp* (6085_h) and subsequent state change to *Quick Stop Active*; control does not switch off and the motor remains energized. You can switch back to the *Operation enabled* state.

The *Quick stop active* state can also be reached when a limit switch is actuated; see Limitation of the range of motion.

5.2.2.3 Ready to switch on

Transition to the *Ready to switch on* state (shutdown option):

In this case, the action stored in object $605B_h$ is executed (see following table).

Value in object $605B_h$	Description
-32768 … -1	Reserved
0	Switch off driver without deceleration ramp; drive function blocked – motor can turn freely
1	Braking with *slow down ramp* (braking deceleration depending on operating mode) and subsequent state change to *Ready to switch on*
2 … 32767	Reserved

5.2.2.4 Switched on

Transition to the *Switched on* state (disable operation option):

In this case, the action stored in object $605C_h$ is executed (see following table).

Value in object $605C_h$	Description
-32768 … -1	Reserved
0	Switch off driver without deceleration ramp; drive function blocked
1	Braking with *slow down ramp* (braking deceleration depending on operating mode) and subsequent state change to *Switched on*
2 … 32767	Reserved

5.2.2.5 Halt

The bit is valid in the following modes:

- Profile Position
- Velocity
- Profile Velocity
- Profile Torque
- Interpolated Position Mode

When setting bit 8 in object 6040_h (controlword), the action stored in $605D_h$ is executed (see following table):

Value in object $605D_h$	Description
-32768 … 0	Reserved
1	Braking with *slow down ramp* (braking deceleration depending on operating mode)
2	Braking with *quick stop ramp* (6085_h)

Value in object $605D_h$	Description
3 … 32767	Reserved

5.2.2.6 Fault

Case of an error (fault):

If an error occurs, the motor will brake according to the value stored in object $605E_h$.

Value in object $605E_h$	Description
-32768 … -1	Reserved
0	Switch off driver without deceleration ramp; drive function blocked – motor can turn freely
1	Braking with *slow down ramp* (braking deceleration depending on operating mode)
2	Braking with *quick stop ramp* (6085_h)
3 … 32767	Reserved

For each error that occurs, a more precise error code is stored in object 1003_h.

5.2.2.7 Following/slippage error

If a following or slippage error occurs, the motor is braked according to the value stored in object 3700_h.

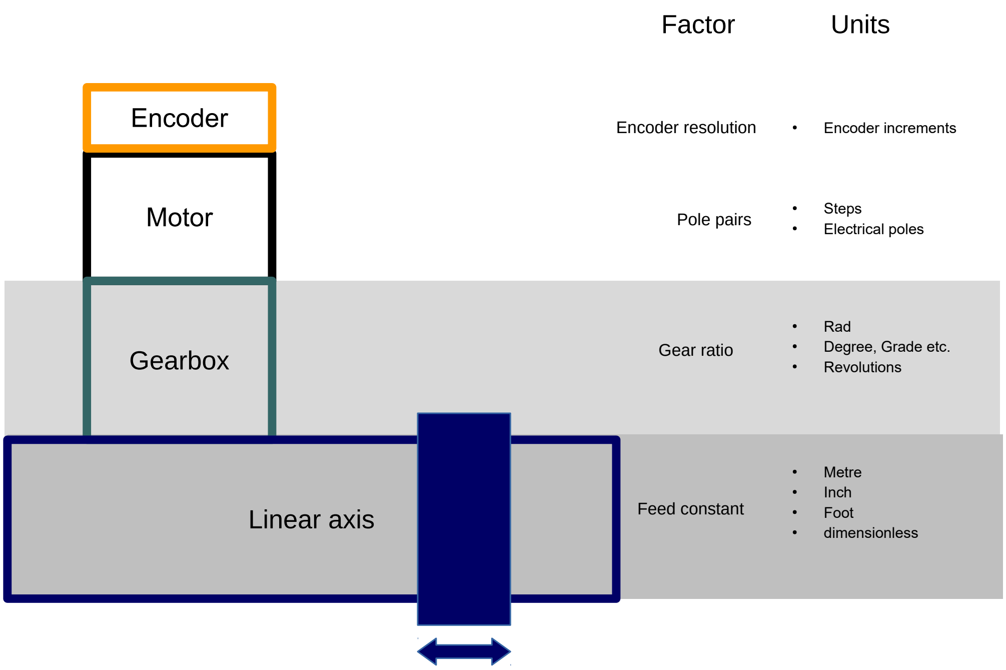
Value	Description
-32768 … -2	Reserved
-1	no reaction
0	Switch off driver without deceleration ramp; drive function blocked – motor can turn freely
1	Braking with *slow down ramp* (braking deceleration depending on operating mode)
2	Braking with *quick stop ramp* (6085_h)
3 … 32767	reserved

You can deactivate error monitoring by setting object 6065_h to the value "-1" ($FFFFFFFF_h$) or object $60F8_h$ to the value "$7FFFFFFF_h$".

5.3 User-defined units

The controller offers you the possibility to set user-defined units. It is thereby possible to set and read out the corresponding parameters, e.g., directly in degrees [°], millimeter [mm], etc.

Depending on the mechanical circumstances, you can also define a Gear ratio and/or a Feed constant.



NOTICE

Value changes of all objects that are described in this chapter are not immediately applied in the *Operation enabled* state of the CiA 402 Power State Machine. For this to happen, the *Operation enabled* state must be exited.

5.3.1 Units

Units of the international unit system (*SI*) as well as a number of specific units are supported. It is also possible to specify a power of ten as a factor.

Listed in the following table are all supported units for the position and their values for $60A8_h$ (Position unit) or $60A9_h$ (Speed unit). Depending on the unit that is used, Feed constant (6092_h) and/or Gear ratio (6091_h) are/is taken into account.

Name	Unit symbol	Value	6091_h	6092_h	Description
meter	m	01_h	yes	yes	*Meter*
inch	in	$C1_h$	yes	yes	*Inch* (=0.0254 m)
foot	ft	$C2_h$	yes	yes	*Foot* (=0.3048 m)
grade	g	40_h	yes	no	*Gradian* (unit of angle, 400 corresponds to 360°)
radian	rad	10_h	yes	no	*Radian*
degree	°	41_h	yes	no	*Degrees*
arcminute	'	42_h	yes	no	*Arcminute* (60'=1°)
arcsecond	"	43_h	yes	no	*Arcsecond* (60"=1')
mechanical revolution		$B4_h$	yes	no	*Revolution*

Name	Unit symbol	Value	6091_h	6092_h	Description
encoder increment		$B5_h$	no	no	*Encoder increments.* Dependent on the used sensor (encoder/Hall sensor) and control mode. In *open-loop* and *sensorless* mode, the number of pole pairs (2030_h) multiplied by 65536 corresponds to one motor revolution.
step		AC_h	no	no	*Steps.* With 2-phase stepper motors, the number of pole pairs (2030_h) multiplied by 4 is equivalent to one revolution. With 3-phase BLDC motors, the number of pole pairs (2030_h) multiplied by 6 is equivalent to one revolution.
electrical pole		$C0_h$	no	no	*Electric poles.* With a stepper motor that has, e.g., 50 pole pairs (2030_h), the unit corresponds to 1/50 of a revolution.
dimensionless		00_h	yes	yes	*Dimensionless length unit*

Listed in the following table are all supported units for the time and their values for $60A9_h$ (Speed unit):

Name	Unit symbol	Value	Description
second	s	03_h	*Second*
minute	min	47_h	*Minute*
hour	h	48_h	*Hour*
day	d	49_h	*Day*
year	a	$4A_h$	*Year* (=365.25 days)

Listed in the following table are the possible exponents and their values for $60A8_h$ (Position unit) and $60A9_h$ (Speed unit):

Factor	Exponent	Value
10^6	6	06_h
10^5	5	05_h
...	...	...
10^1	1	01_h
10^0	0	00_h
10^{-1}	-1	FF_h
...	..	...
10^{-5}	-5	FB_h
10^{-6}	-6	FA_h

5.3.2 Encoder resolution

The physical resolution for position measurement of the used encoder/sensor is calculated from the encoder increments ($60E6_h$ (Encoder Increments)) per motor revolutions ($60EB_h$ (Motor Revolutions)).

5.3.3 Gear ratio

The gear ratio is calculated from motor revolutions ($60E8_h$ (Motor Shaft Revolutions)) per axis rotations ($60ED_h$ (Driving Shaft Revolutions)).

5.3.4 Feed constant

The feed constant is calculated in user-defined position units from the feed (60E9$_h$ (Feed) per revolution of the output shaft (60EE$_h$ (Driving Shaft Revolutions).

The feed constant is useful for specifying the lead screw pitch for a linear axis and is used if the unit is based on length dimensions or if it is dimensionless.

5.3.5 Calculation formulas for user units

5.3.5.1 Position unit

Object 60A8$_h$ contains:

- Bits 16 to 23: The position unit (see chapter Units)
- Bits 24 to 31: The exponent of a power of ten (see chapter Units)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Factor								Unit							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
reserved (00h)								reserved (00h)							

Example

If 60A8$_h$ is written with the value "FF410000$_h$" (bits 16-23=41$_h$ and bits 24-31=FF$_h$), the unit is set to *tenths of degree* (factory setting).

With a relative target position (607A$_h$) of 3600, the motor moves exactly one mechanical revolution, if Gear ratio is 1:1. The Feed constant plays no role in this case.

Example

If 60A8$_h$ is written with the value "FD010000$_h$" (bits 16-23=01$_h$ and bits 24-31=FD$_h$(=-3)), the unit is set to *millimeter*.

With a relative target position (607A$_h$) of 1, the motor moves exactly one mechanical revolution, if Feed constant and Gear ratio are 1:1.

If the Feed constant is set according to the lead screw pitch of a linear axis, the motor turns far enough that a feed of 1 mm is achieved.

5.3.5.2 Speed unit

Object 60A9$_h$ contains:

- Bits 8 to 15: The time unit (see chapter Units)
- Bits 16 to 23: The position unit (see chapter Units)
- Bits 24 to 31: The exponent of a power of ten (see chapter Units)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Factor								Nominator (Position)							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Denominator (Time)								reserved (00h)							

Example

If 60A9$_h$ is written with the value "00B44700$_h$" (bits 8-15=00$_h$, bits 16-23=B4$_h$ and bits 24-31=47$_h$), the unit is set to *revolutions per minute* (factory setting).

Example

If 60A9$_h$ is written with the value "FD010300$_h$" (bits 8-15=FD$_h$(=-3), bits 16-23=01$_h$ and bits 24-31=03$_h$), the unit is set to *millimeters per second*.

NOTICE

The speed unit in Velocity mode is preset to *revolutions per minute*. You can only set the unit via the 604Ch Vl Dimension Factor.

Conversion factor for the speed unit

You can set an additional factor for the speed unit. Thus, a unit of, e.g., 1/3 revolutions/minute is possible. The factor n is calculated from the factor for numerator (6096$_h$:01$_h$) divided by the factor for denominator (6096$_h$:02$_h$).

$$n_{velocity} = \frac{6096_h{:}01}{6096_h{:}02}$$

5.3.5.3 Acceleration unit

The acceleration unit is speed unit per second.

Conversion factor for the acceleration unit

The factor n for the acceleration unit is calculated from the numerator (6097$_h$:01$_h$) divided by the denominator (6097$_h$:02$_h$).

$$n_{acceleration} = \frac{6097_h{:}01}{6097_h{:}02}$$

5.3.5.4 Jerk unit

The jerk unit is Acceleration unit per second.

Conversion factor for jerk

The factor n for the jerk is calculated from the numerator (60A2$_h$:01$_h$) divided by the denominator (60A2$_h$:02$_h$).

$$n_{jerk} = \frac{60A2_h{:}01}{60A2_h{:}02}$$

5.4 Limitation of the range of motion

5.4.1 Software limit switches

The controller takes into account software limit switches ($607D_h$ (Software Position Limit)). Target positions ($607A_h$) are limited by $607D_h$; the absolute target position may not be larger than the limits in $607D_h$. If the motor is located outside of the permissible range when setting up the limit switches, only travel commands in the direction of the permissible range are accepted.

5.5 Cycle times

The controller operates with a cycle time of 1 ms. This means that data are processed every 1 ms; multiple changes to a value (e.g., value of an object or level at a digital input) within one ms cannot be detected.

The following table includes an overview of the cycle times of the various processes.

Task	Cycle time
Application	1 ms
NanoJ application	1 ms
Current controller	62.5 µs (16 KHz)
Velocity controller	250 µs (4 kHz)
Position controller	1 ms

6 operating modes

6.1 Profile Position

6.1.1 Overview

6.1.1.1 Description

Profile Position Mode is used to move to positions relative to the last target position or to an absolute position (last reference position). During the movement, the limit values for the speed, starting acceleration/ braking deceleration and jerks are taken into account.

6.1.1.2 Activation

To activate the mode, the value "1" must be set in object 6060_h (Modes Of Operation) (see "CiA 402 Power State Machine").

6.1.1.3 Controlword

The following bits in object 6040_h (controlword) have a special function:

- Bit 4 starts a travel command. This is carried out on a transition from "0" to "1". An exception occurs if changing from another operating mode to *profile position*: If bit 4 is already set, it does not need to be set to "0" and then back to "1" in order to start the travel command.
- Bit 5: If this bit is set to "1", a travel command triggered by bit 4 is immediately executed. If it is set to "0", the just executed travel command is completed and only then is the next travel command started.
- Bit 6: With "0", the target position ($607A_h$) is absolute and with "1" the target position is relative. The reference position is dependent on bits 0 and 1 of object $60F2_h$.
- Bit 8 (Halt): If this bit is set to "1", the motor stops. On a transition from "1" to "0", the motor accelerates with the set start ramp to the target speed. On a transition from "0" to "1", the motor brakes and comes to a standstill. The braking deceleration is dependent here on the setting of the "Halt Option Code" in object $605D_h$.
- Bit 9 (Change on setpoint): If this bit is set, the speed is not changed until the first target position is reached. This means that, before the first target is reached, no braking is performed, as the motor should not come to a standstill at this position.

		Controlword 6040_h
Bit 9	**Bit 5**	**Definition**
X	1	The new target position is moved to immediately.
0	0	Positioning is completed before moving to the next target position with the new limits.
1	0	The current target position is only passed through; afterwards, the new target position is moved to with the new values.

For further information, see figure in "Setting travel commands".

NOTICE

Bit 9 in the controlword is ignored if the ramp speed is not met at the target point. In this case, the controller would need to reset and take a run-up to reach the preset.

6.1.1.4 Statusword

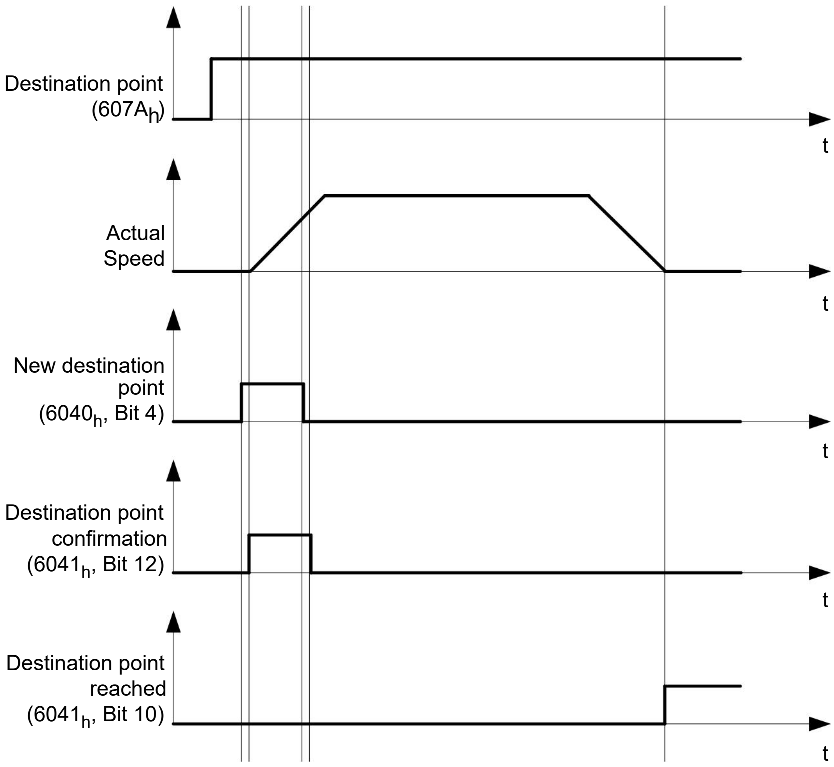
The following bits in object 6041_h (statusword) have a special function:

- Bit 10 (Target Reached): This bit is set to "1" if the last target was reached and the motor remains within a tolerance window (6067$_h$) for a preset time (6068$_h$). The bit is also set to "1" if the halt bit (bit 8) in 6040$_h$ has been set and as soon as the motor is at a standstill.
- Bit 11: Limit exceeded: The demand position is above or below the limit values set in 607D$_h$.
- Bit 12 (Set-point acknowledge): This bit confirms receipt of a new and valid set point. It is set and reset in sync with the "New set-point" bit in the controlword.
  There is an exception in the event that a new movement is started before another one has completed and the next movement is not to occur until after the first one has finished. In this case, the bit is reset if the command was accepted and the controller is ready to execute new travel commands. If a new travel command is sent even though this bit is still set, the newest travel command is ignored.
  The bit is not set if one of the following conditions is met:
  - The new target position can no longer be reached while adhering to all boundary conditions.
  - A target position was already traveled to and a target position was already specified. A new target position can only be specified after the current positioning has been concluded.
- Bit 13 (Following Error): This bit is set in *closed loop* mode if the following error is greater than the set limits (6065$_h$ (Following Error Window) and 6066$_h$ (Following Error Time Out)).

6.1.2 Setting travel commands

6.1.2.1 Travel command

In object 607A$_h$ (Target Position), the new target position is specified in user units (see User-defined units). The travel command is then triggered by setting bit 4 in object 6040$_h$ (controlword). If the target position is valid, the controller responds with bit 12 in object 6041$_h$ (statusword) and begins the positioning move. As soon as the position is reached, bit 10 in the statusword is set to "1".



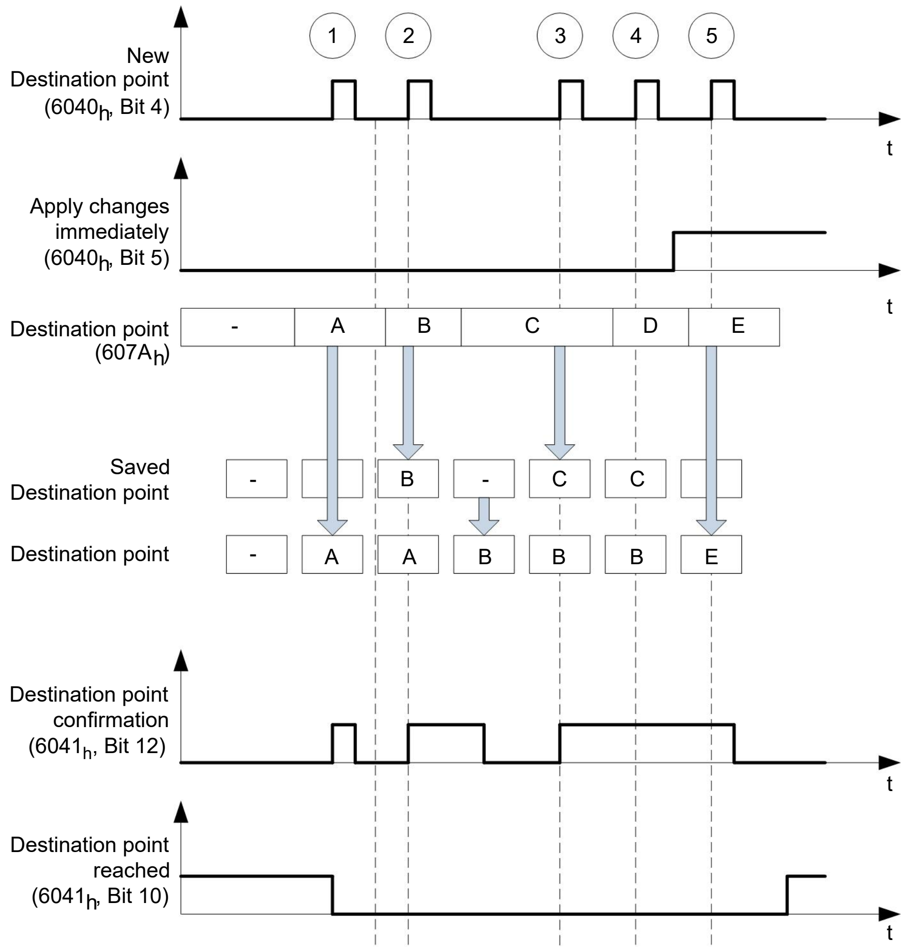
The controller can also reset bit 4 in object 6040$_h$ (controlword) on its own. This is set with bits 4 and 5 of object 60F2$_h$.

6.1.2.2 Other travel commands

Bit 12 in object 6041$_h$ (statusword, set-point acknowledge) changes to "0" if another travel command can be buffered (see time 1 in the following figure). As long as a target position is being moved to, a second target position can be passed to the controller in preparation. All parameters – such as speed, acceleration, braking deceleration, etc. – can thereby be reset (time 2). If the buffer is empty, the next time can be queued up (time 3).

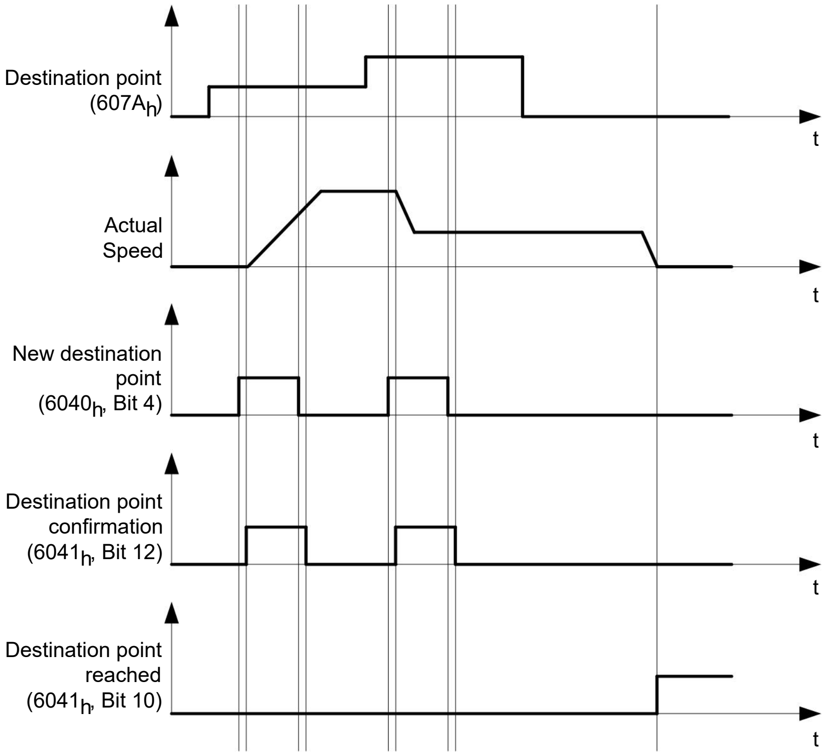
If the buffer is already full, a new set point is ignored (time 4). If bit 5 in object 6040_h (controlword, bit: "Change Set-Point Immediately") is set, the controller operates without the buffer; new travel commands are implemented directly (time 5).

Times



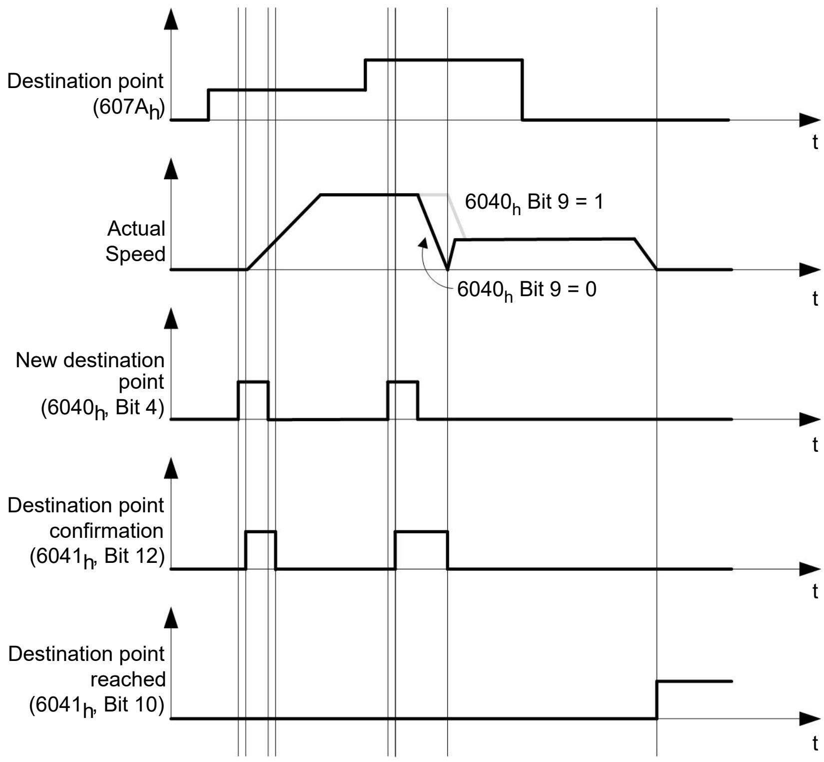
Transition procedure for second target position

The following graphic shows the transition procedure for the second target position while moving to the first target position. In this figure, bit 5 of object 6040_h (controlword) is set to "1"; the new target value is, thus, taken over immediately.



Possibilities for moving to a target position

If bit 9 in object 6040h (controlword) is equal to "0", the current target position is first moved to completely. In this example, the final speed (6082h) of the target position is equal to zero. If bit 9 is set to "1", the profile speed (6081h) is maintained until the target position is reached; only then do the new boundary conditions apply.

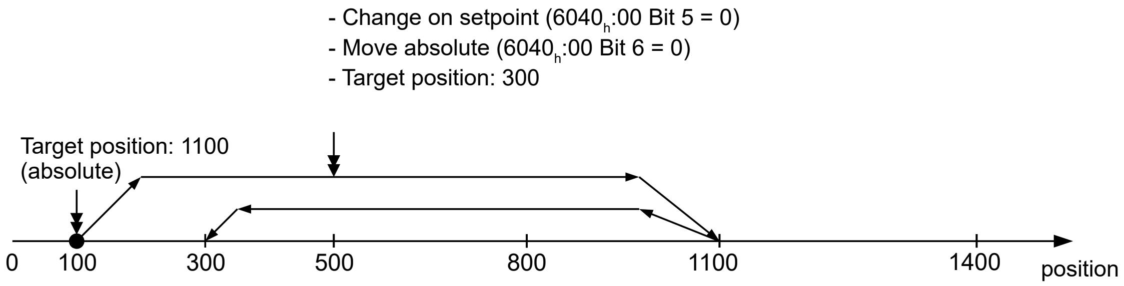


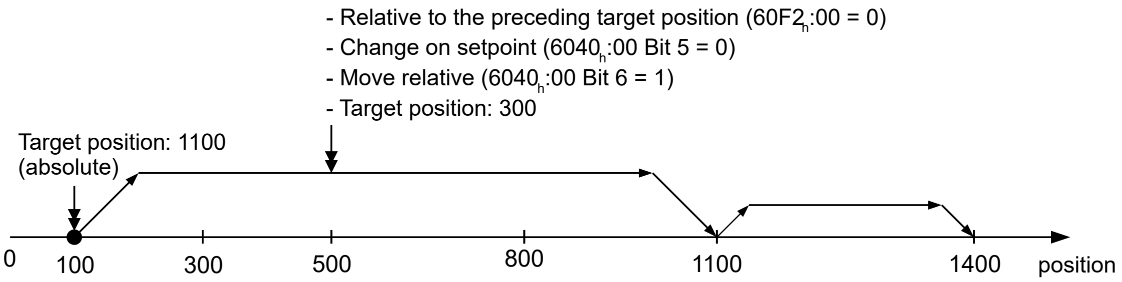
Possible combinations of travel commands

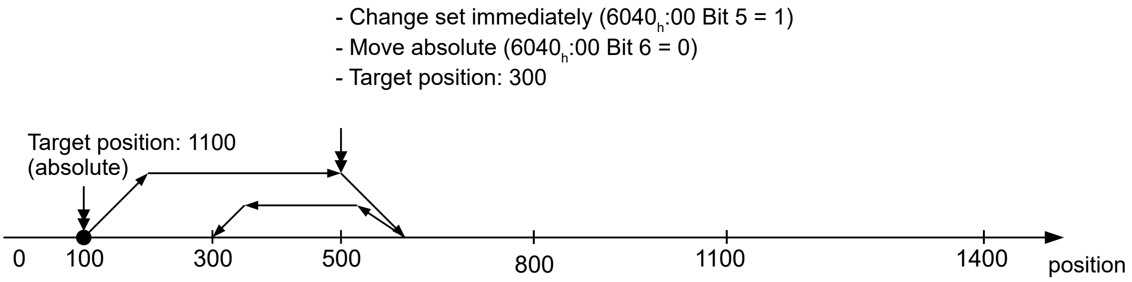
To provide a better overview of the travel commands, combinations of travel commands are listed and depicted in this chapter.

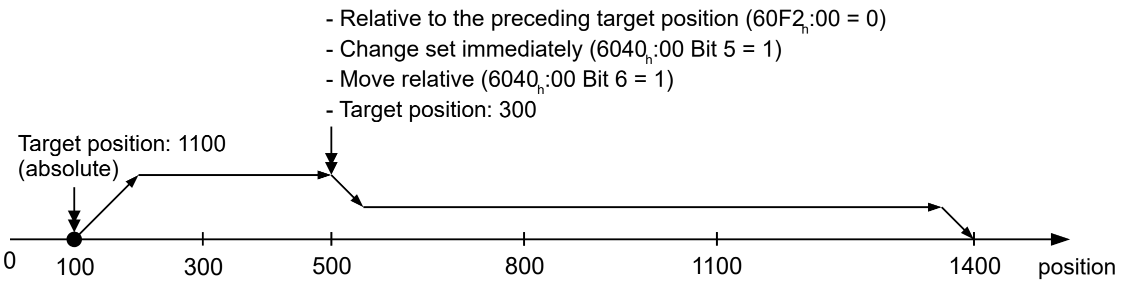
The following applies for the figures below:

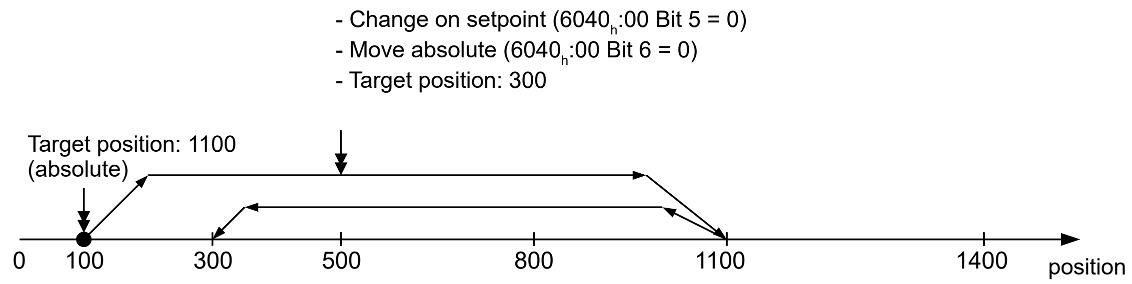
- A double arrow indicates a new travel command.
- The first travel command at the start is always an absolute travel command to position 1100.
- The second movement is performed at a lower speed so as to present the graphs in a clear manner.

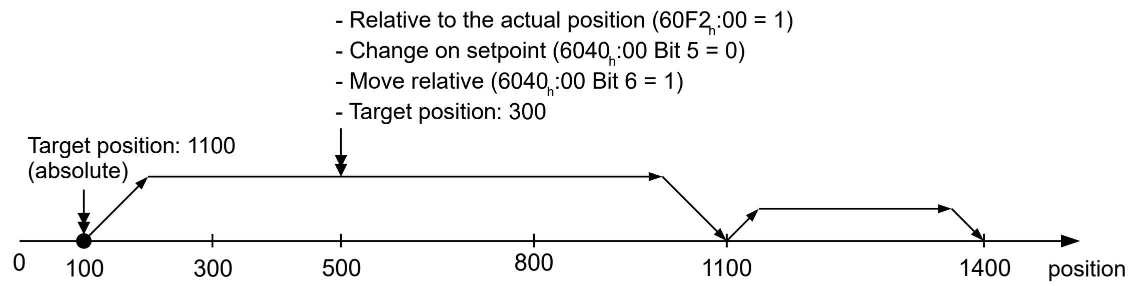


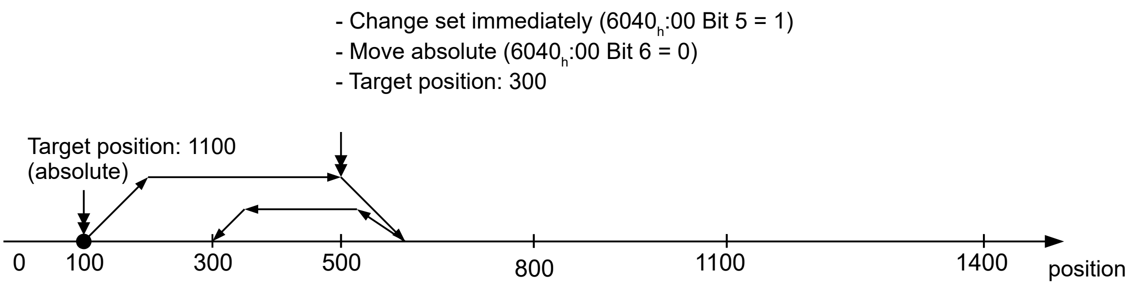






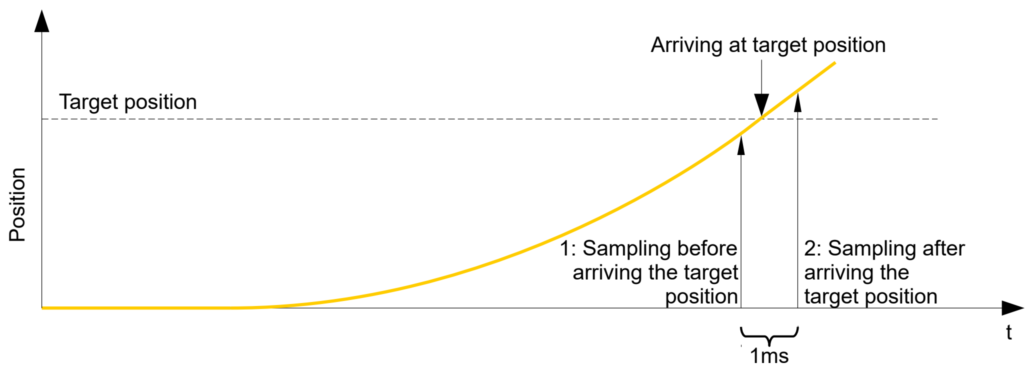






6.1.3 Loss of accuracy for relative movements

When linking together relative movements, a loss of accuracy may occur if the final speed is not set to zero. The following graphic illustrates the reason.



The current position is sampled once per millisecond. It is possible that the target position is reached between two samples. If the final speed is not equal to zero, then, after the target position is reached, the sample is used as an offset as the basis for the subsequent movement. As a result, the subsequent movement may go somewhat farther than expected.

6.1.4 Boundary conditions for a positioning move

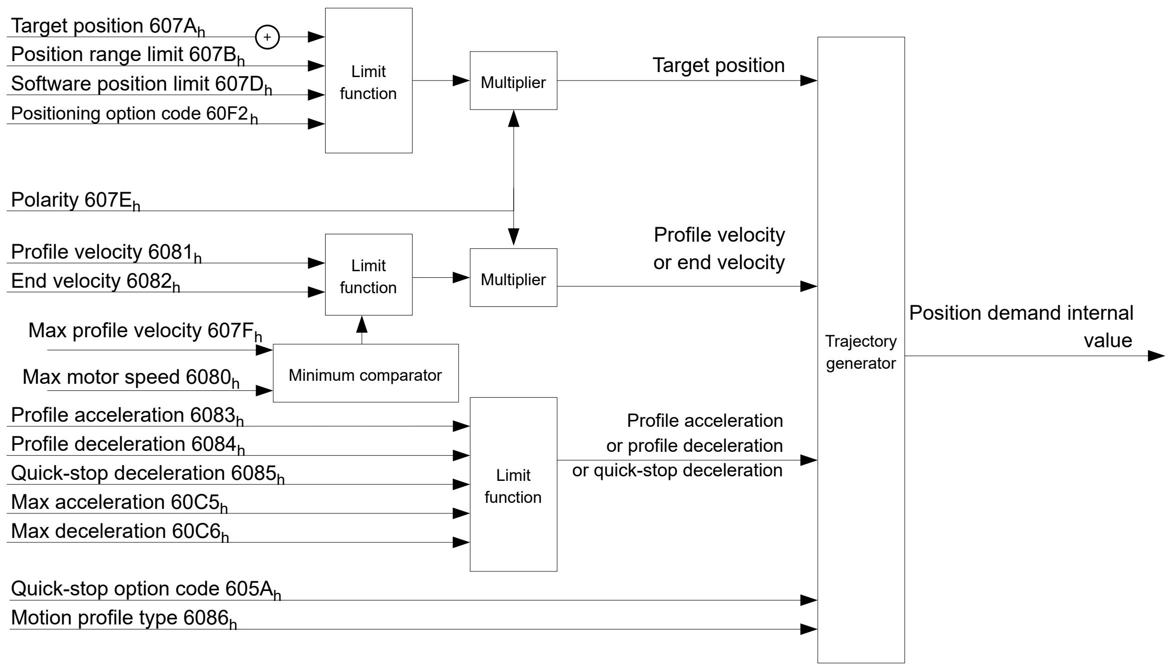
6.1.4.1 Object entries

The boundary conditions for the position that has been moved to can be set in the following entries of the object dictionary:

- 607A$_h$: (Target Position): Planned target position
- 607D$_h$: (Software Position Limit): Definition of the limit stops (see chapter Software limit switches)
- 607C$_h$ (Home Offset): Specifies the difference between the zero position of the controller and the reference point of the machine in user-defined units. (See "Homing")
- 607B$_h$ (Position Range Limit): Limits of a modulo operation for replicating an endless rotation axis
- 607$_h$ (Polarity): Direction of rotation
- 6081$_h$ (Profile Velocity): Maximum speed with which the position is to be approached
- 6082$_h$ (End Velocity): Speed upon reaching the target position
- 6083$_h$ (Profile Acceleration): Desired starting acceleration
- 6084$_h$ (Profile Deceleration): Desired braking deceleration
- 6085$_h$ (Quick Stop Deceleration): Emergency-stop braking deceleration in case of the "Quick stop active" state of the "CiA 402 Power State Machine"
- 6086$_h$ (Motion Profile Type): Type of ramp to be traveled; if the value is "0", the jerk is not limited; if the value is "3", the values of 60A4$_h$:1$_h$–4$_h$ are set as limits for the jerk.
- 60C5$_h$ (Max Acceleration): The maximum acceleration that may not be exceeded when moving to the end position
- 60C6$_h$ (Max Deceleration): The maximum braking deceleration that may not be exceeded when moving to the end position
- 60A4$_h$ (Profile Jerk), subindex 01$_h$ to 04$_h$: Objects for specifying the limit values for the jerk.
- The speed is is limited by 607F$_h$ (Max Profile Velocity) and 6080$_h$ (Max Motor Speed); the smaller value is used as the limit.
- 60F2$_h$: (Positioning Option Code): Defines the positioning behavior

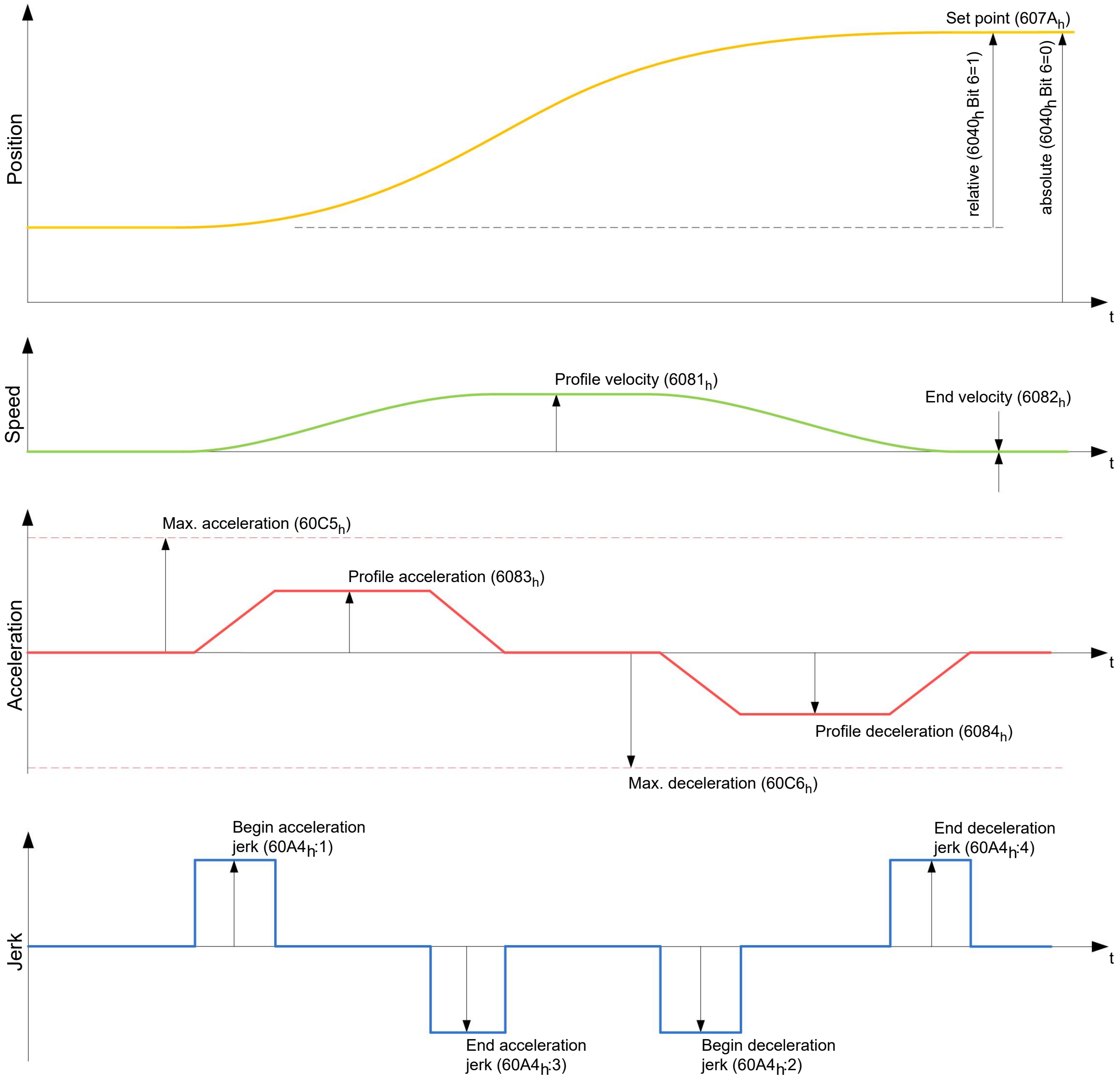
6.1.4.2 Objects for the positioning move

The following graphic shows the objects involved in the boundary conditions of the positioning move.



6.1.4.3 Parameters for the target position

The following graphic shows an overview of the parameters that are used for moving to a target position (figure not to scale).



6.1.5 Jerk-limited mode and non-jerk-limited mode

6.1.5.1 Description

A distinction is made between the "jerk-limited" and "non-jerk-limited" modes.

6.1.5.2 Jerk-limited mode

Jerk-limited positioning can be achieved by setting object 6086$_h$ to "3". The entries for the jerks in subindices :1$_h$–4$_h$ of object 60A4 thereby become valid.

6.1.5.3 Non-jerk-limited mode

A "non-jerk-limited" ramp is traveled if the entry in object 6086$_h$ is set to "0" (default setting).

6.2 Velocity

6.2.1 Description

This mode operates the motor at a preset target speed, similar to a frequency inverter. Unlike the *profile velocity mode*, this mode does not permit the selection of jerk-limited ramps.

6.2.2 Activation

To activate the mode, the value "2" must be set in object 6060_h (Modes Of Operation) (see "CiA 402 Power State Machine").

6.2.3 Controlword

The following bits in object 6040_h (controlword) have a special function:

- Bit 8 (Halt): If this bit is set to "1", the motor stops. On a transition from "1" to "0", the motor accelerates with the acceleration ramp to the target speed. On a transition from "0" to "1", the motor brakes according to the deceleration ramp and comes to a standstill.

6.2.4 Statusword

The following bits in object 6041_h (statusword) have a special function:

- Bit 11: Limit exceeded: The target speed is above or below the set limit values.

6.2.5 Object entries

The following objects are necessary for controlling this mode:

- $604C_h$ (Dimension Factor):
  The unit for speed values is defined here for the following objects.
  Subindex 1 contains the denominator (multiplier) and subindex 2 contains the numerator (divisor) with which the internal speed values are converted to revolutions per minute. If, for example, subindex 1 is set to the value "60" and subindex 2 is set to the value "1", the speed is specified in revolutions per second (60 revolutions per 1 minute).
- 6042_h: Target Velocity.
  The target speed is set here in user-defined units.
- 6048_h: Velocity Acceleration
  This object defines the acceleration. Subindex 1 contains the change in speed, subindex 2 the corresponding time in seconds. Both together are used to calculate the acceleration:

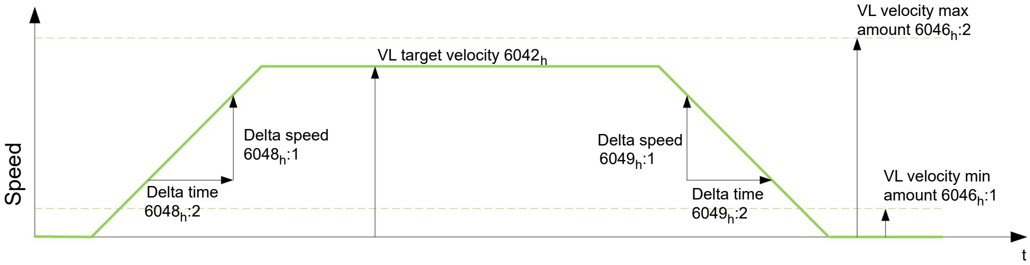
$$\text{VL velocity acceleration} = \frac{\text{Delta speed } (6048_h{:}1)}{\text{Delta time } (6048_h{:}2)}$$

- 6049_h (Velocity Deceleration):
  This object defines the deceleration (deceleration ramp). The subindices here are arranged as described in object 6048_h; the change in speed is to be specified with positive sign.
- 6046_h (Velocity Min Max Amount):
  The limitations of the target speeds are specified in this object.
  The minimum speed is set in $6046_h{:}1_h$. If the target speed (6042_h) falls below the minimum speed, the value is limited to the minimum speed $6046_h{:}1_h$.
  The maximum speed is set in $6046_h{:}2_h$. If the target speed (6042_h) exceeds the maximum speed, the value is limited to the maximum speed $6046_h{:}2_h$.
- $604A_h$ (Velocity Quick Stop):
  This object can be used to set the quick-stop ramp. Subindices 1 and 2 are identical to those described for object 6048_h.

The following objects can be used to check the function:

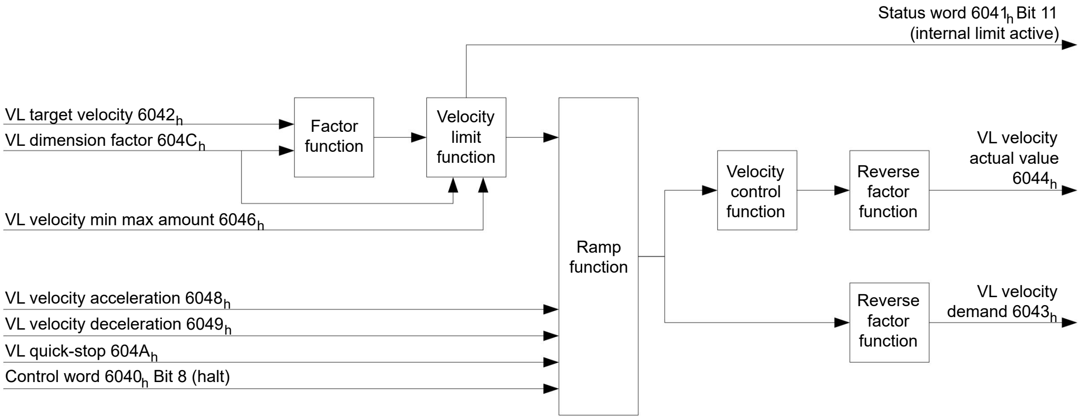
- 6043_h (Vl Velocity Demand)
- 6044_h (Vl Velocity Actual Value)

6.2.5.1 Speeds in Velocity Mode



6.2.5.2 Objects for Velocity Mode

The ramp generator follows the target speed, remaining within the set speed and acceleration limits. As long as a limit is active, bit 11 in object 6041h is set (internal limit active).



6.3 Profile Velocity

6.3.1 Description

This mode operates the motor in Velocity Mode with extended (jerk-limited) ramps. Unlike *Velocity Mode* (see "Velocity"), the statusword is used in this mode to indicate whether the target speed is reached.

6.3.2 Activation

To activate the mode, the value "3" must be set in object 6060h (Modes Of Operation) (see "CiA 402 Power State Machine").

6.3.3 Controlword

The following bits in object 6040h (controlword) have a special function:

- Bit 8 (Halt): If this bit is set to "1", the motor stops. On a transition from "1" to "0", the motor accelerates with the set start ramp to the target speed. On a transition from "0" to "1", the motor brakes and comes to a standstill.

6.3.4 Statusword

The following bits in object 6041h (statusword) have a special function:

- Bit 10 (target speed reached; Target Reached): In combination with bit 8 in the controlword, this bit specifies whether the target speed is reached, if braking is taking place or if the motor is at a standstill (see table).

6041$_h$ Bit 10	6040$_h$ Bit 8	Description
0	0	Target speed not reached
0	1	Axis braking
1	0	Target speed within target window (defined in 606D$_h$h and 606E$_h$)
1	1	Axis speed is 0

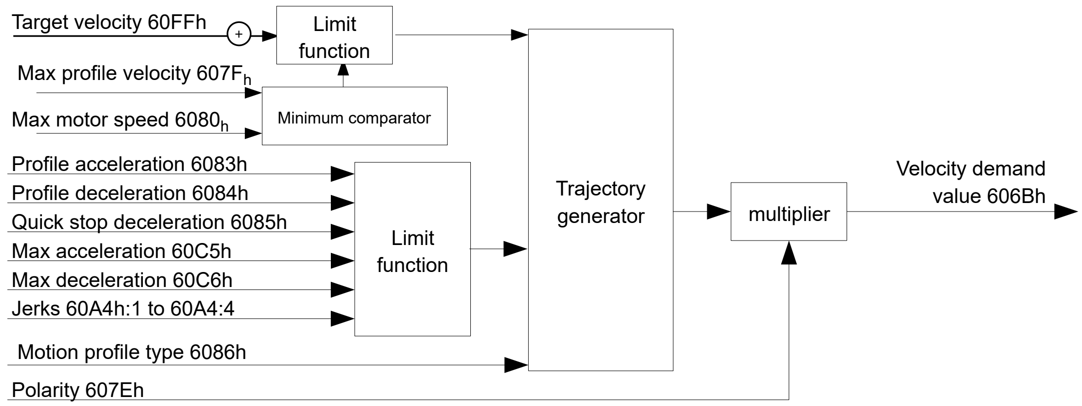
- Bit 12: This bit indicates whether the actual speed is zero.
  If the actual speed is greater than the value in 606F$_h$(Velocity Threshold) for a time of 6070$_h$(Velocity Threshold Time), this bit has the value "0". The bit otherwise remains set to "1".
- Bit 13 (Deviation Error): This bit is set in *closed loop* mode if the slippage error is greater than the set limits (60F8h Max Slippage and 203Fh Max Slippage Time Out).

6.3.5 Object entries

The following objects are necessary for controlling this mode:

- 606B$_h$ (Velocity Demand Value):
  This object contains the output of the ramp generator, which simultaneously serves as the preset value for the velocity controller.
- 606C$_h$ (Velocity Actual Value):
  Indicates the current actual speed.
- 606D$_h$ (Velocity Window):
  This value specifies by how much the actual speed may vary from the set speed for bit 10 (target speed reached; Target Reached") in object 6041$_h$ (statusword) to be set to "1".
- 606E$_h$ (Velocity Window Time):
  This object specifies how long the actual speed and the set speed must be close to one another (see 606D$_h$ "Velocity Window") for bit 10 "Target speed reached" in object 6041$_h$ (statusword) to be set to "1".
- 607E$_h$ (Polarity):
  If bit 6 is set to "1" here, the sign of the target speed is reversed.
- 6083$_h$ (Profile acceleration):
  Sets the value for the acceleration ramp.
- 6084$_h$ (Profile Deceleration):
  Sets the value for the deceleration ramp.
- 6085$_h$ (Quick Stop Deceleration):
  Sets the value for the deceleration ramp for rapid braking.
- 6086$_h$ (Motion Profile Type):
  The ramp type can be selected here ("0" = trapezoidal ramp, "3" = jerk-limited ramp).
- 60FF$_h$ (Target Velocity):
  Specifies the target speed that is to be reached.
- The speed is is limited by 607F$_h$ (Max Profile Velocity) and 6080$_h$ (Max Motor Speed); the smaller value is used as the limit.

6.3.5.1 Objects in Profile Velocity Mode

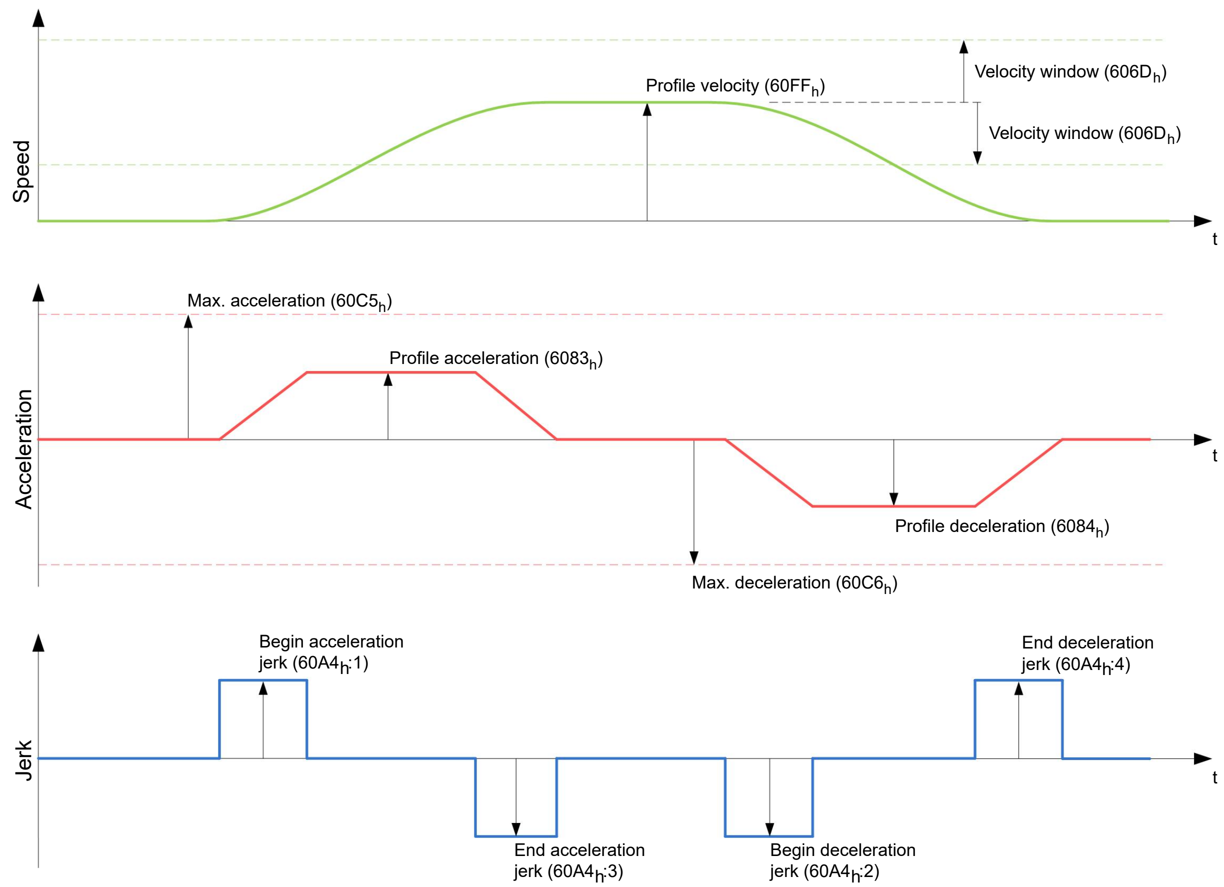


6.3.5.2 Activation

After the mode is selected in object 6060h (Modes Of Operation) and the "Power State machine" (see "CiA 402 Power State Machine") is switched to *Operation enabled*, the motor is accelerated to the target speed in object 60FFh (see following figures). The speed and acceleration values are taken into account here; for jerk-limited ramps, the jerk-limit values are also taken into account.

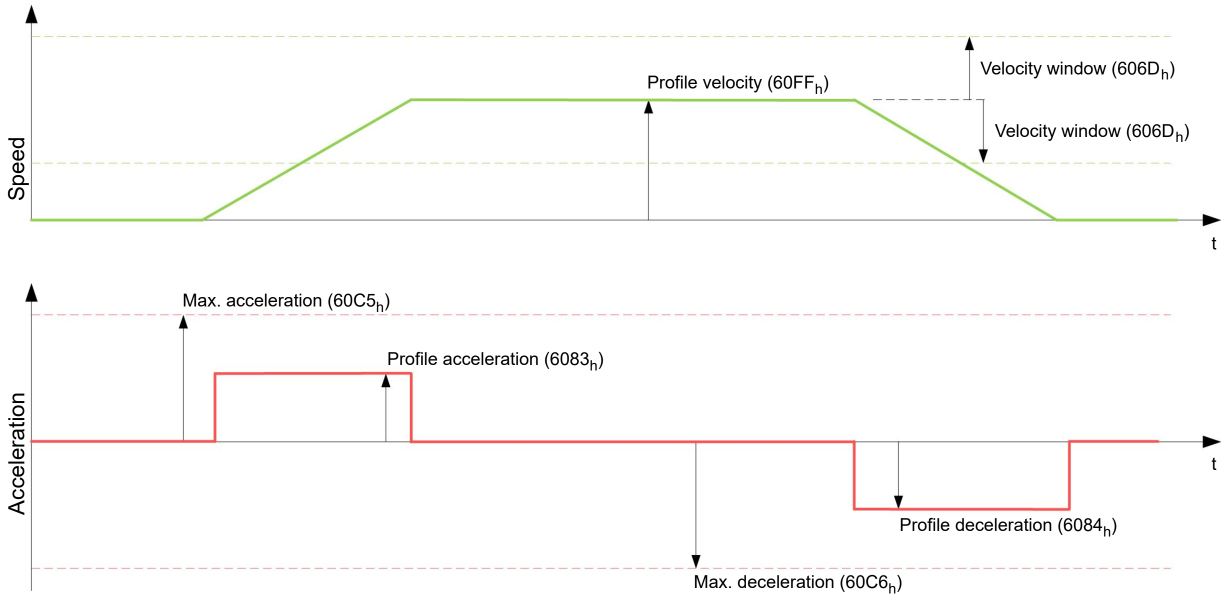
6.3.5.3 Limitations in the jerk-limited case

The following figure shows the adjustable limits in the jerk-limited case (6086h = 3).



6.3.5.4 Limitations in the trapezoidal case

This figure shows the adjustable limitations for the trapezoidal case (6086_h = 0).



6.4 Profile Torque

6.4.1 Description

In this mode, the torque is preset as a set value and reached via a ramp function.

NOTICE

This mode only functions if closed loop is activated, see also Commissioning Closed Loop.

6.4.2 Activation

To activate the mode, the value "4" must be set in object 6060_h (Modes Of Operation) (see "CiA 402 Power State Machine").

6.4.3 Controlword

The following bits in object 6040_h (controlword) have a special function:

- Bit 8 (Halt): If this bit is set to "1", the motor stops. If this bit is set from "1" to "0", the motor is started up according to the presets. When setting from "0" to "1", the motor is again brought to a standstill, taking the preset values into consideration.

6.4.4 Statusword

The following bits in object 6041_h (statusword) have a special function:

- Bit 10 (Target Reached): In combination with bit 8 of object 6040_h (controlword), this bit indicates whether the specified torque is reached (see following table). The target is considered having been met if the current torque (6077h Torque Actual Value) is within a tolerance window (203Dh Torque Window) for a specified time (203Eh Torque Window Time Out).

6040_h Bit 8	6041_h Bit 10	Description
0	0	Specified torque not reached

6040h Bit 8	6041h Bit 10	Description
0	1	Specified torque reached
1	0	Axis brakes
1	1	Axis speed is 0

- Bit 11: Limit exceeded: The target torque (6071_h) exceeds the maximum torque entered in 6072_h.

6.4.5 Object entries

All values of the following entries in the object dictionary are to be specified as a thousandth of the maximum torque, which corresponds to the rated current ($203B_h$:01_h) . This includes the objects:

- 6071_h (Target Torque):
  Target torque
- 6072_h (Max Torque):
  Maximum torque during the entire ramp (accelerate, maintain torque, decelerate)
- 6073_h (Max Current):
  Maximum current. The minimum of 6073_h and 6072_h is used as limit for the torque in 6071_h.
- 6074_h (Torque Demand):
  Current output value of the ramp generator (torque) for the controller
- 6077 (Torque Actual Value):
  Current torque value
- 6087_h (Torque Slope):
  Max. change in torque per second

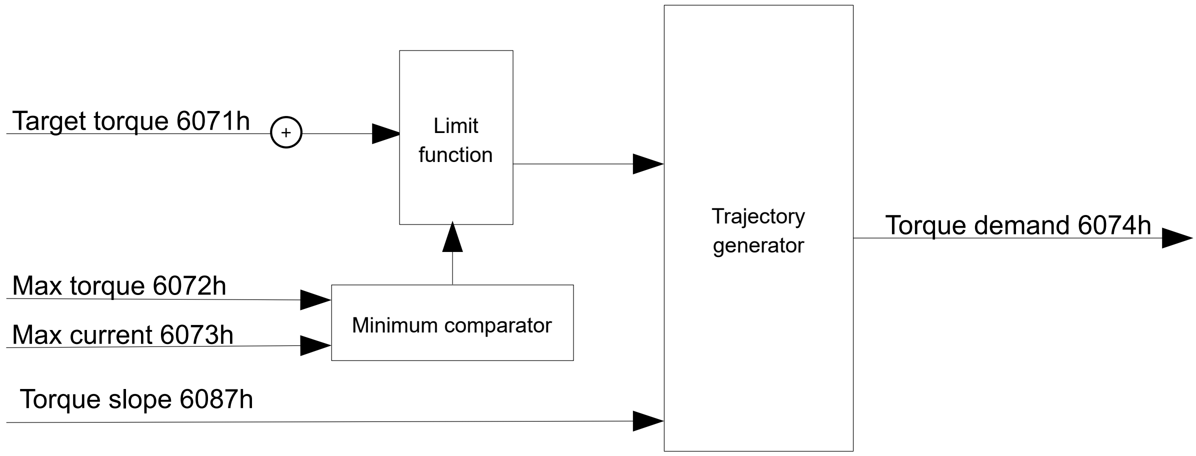
NOTICE

These values are not limited to 100% of the rated current ($203B_h$:01_h). Torque values greater than the rated torque (generated from the rated current) can be achieved if the maximum duration ($203B_h$:02_h) of the maximum current (6073_h) is set (see I2t Motor overload protection). All torque objects are limited by the maximum motor current (2031_h).

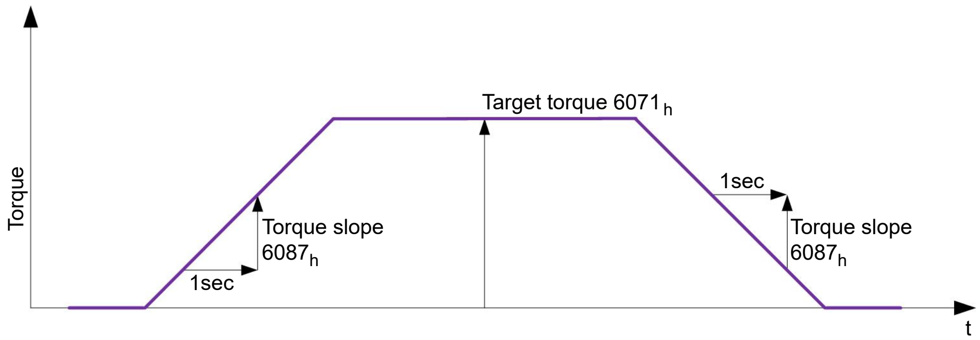
The following objects are also needed for this operating mode:

- 3202_h Bit 5 (Motor Drive Submode Select):
  If this bit is set to "0", the drive controller is operated in the torque-limited Velocity Mode, i.e., the maximum speed can be limited in object 6080_h and the controller can operate in field weakening mode.
  If this bit is set to "1", the controller operates in the ("Real") Torque Mode; the maximum speed cannot be limited here and field weakening mode is not possible.

6.4.5.1 Objects of the ramp generator



6.4.5.2 Torque curve



6.5 Homing

6.5.1 Overview

6.5.1.1 Description

The purpose of the homing method is to align the position zero point of the controller with an encoder index or position switch.

6.5.1.2 Activation

To activate the mode, the value "6" must be set in object 6060_h (Modes Of Operation) (see "CiA 402 Power State Machine").

TIP

If home switches and/or limit switches are used, these special functions must first be activated in the I/O configuration (see "Digital inputs and outputs").

To use the limit switch, you must also set object 3701_h to "-1" (factory setting) to prevent blocking the further travel of the motor.

6.5.1.3 Controlword

The following bits in object 6040_h (controlword) have a special function:

- Bit 4: If the bit is set to "1", referencing is started. This is performed until either the reference position is reached or bit 4 is reset to "0".

6.5.1.4 Statusword

The following bits in object 6041_h (statusword) have a special function:

Bit 13	Bit 12	Bit 10	Description
0	0	0	Homing is performed
0	0	1	Homing is interrupted or not started
0	1	0	Homing has been performed since the last restart but target is not currently reached
0	1	1	Homing completed
1	0	0	Error during homing, motor still turning
1	0	1	Error during homing, motor at standstill

NOTICE

Bit 12 in *Homing* mode is set to 1 after the first fully completed homing operation since the restart. It is only reset to 0

- during all subsequent homing operations
- in the event of an error during a homing operation (permanently deleted until a new homing operation is fully completed).

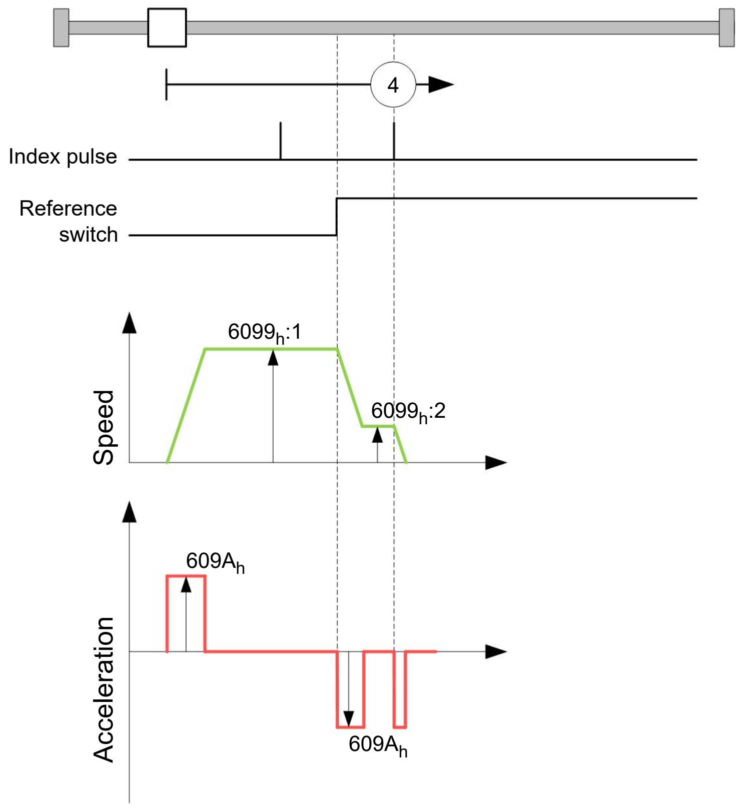
6.5.1.5 Object entries

The following objects are necessary for controlling this mode:

- $607C_h$ (Home Offset): Specifies the difference between the zero position of the controller and the reference point of the machine in user-defined units.
- 6098_h (Homing Method):
  Method to be used for referencing (see "Homing method")
- 6099_h:01_h (Speed During Search For Switch):
  Speed for the search of the switch
- 6099_h:02_h (Speed During Search For Zero):
  Speed for the search of the index
- 6080_h (Max Motor Speed): Maximum speed
- $609A_h$ (Homing Acceleration):
  Starting acceleration and braking deceleration for homing. Also used in case of a quick stop.
- $203A_h$:01_h (Minimum Current For Block Detection):
  Minimum current threshold which, if exceeded, is to detect the blocking of the motor at a block.
- $203A_h$:02_h (Period Of Blocking):
  Specifies the time in ms that the motor is to continue to run against the block after block detection.

Homing speeds

The figure shows the homing speeds using method 4 as an example:



6.5.2 Homing method

6.5.2.1 Description

The homing method is written as a number in object 6098h and decides whether, on a switch edge (rising/ falling), a current threshold for block detection or an index pulse is referenced or in which direction homing starts. Methods that use the index pulse of the encoder lie in the number range 1 to 14, 33 and 34. Methods that do not use the index pulse of the encoder lie between 17 and 30, but are identical to methods 1 to 14 with respect to the travel profiles. These number are shown in circles in the following figures. Methods for which no limit switches are used and, instead, travel against a block is to be detected, a minus must be placed before the method number when making the call.

In the following graphics, the negative movement direction is to the left. The *limit switch* is located before the respective mechanical block; the *home switch* is located between the two limit switches. The index pulses come from the connected encoder.

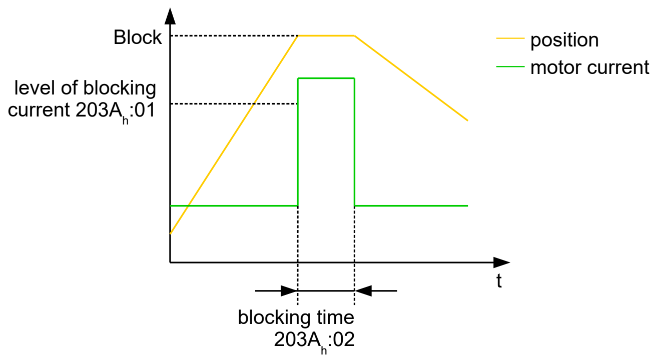
For methods that use homing on block, the same figures apply as for the methods with limit switch. Because nothing is different aside from the missing limit switches, the same figures are used. For the figures here, the limit switches must be replaced with a mechanical block.

6.5.2.2 Homing on block

Homing on block currently only functions in *closed loop* mode.

"Homing on block" functions like every homing method with the difference that instead of a limit switch, a block (limit stop) is used for positioning. Two settings are to be made here:

1. Current level: In object 203Ah:01, the current level is defined above which movement against the block is detected.
2. Blocking duration: In object 203Ah:02, the duration during which the motor moves against the block is set.



6.5.2.3 Overview of methods

Methods 1 to 14 as well as 33 and 34 use the index pulse of the encoder.

Methods 17 to 32 are identical to methods 1 to 14 with the difference that only limit or home switches are used for referencing and not the index pulse.

- Methods 1 to 14 use an index pulse.
- Methods 17 to 30 do not use an index pulse.
- Methods 33 and 34 reference only to the next index pulse.
- Method 35 references to the current position.

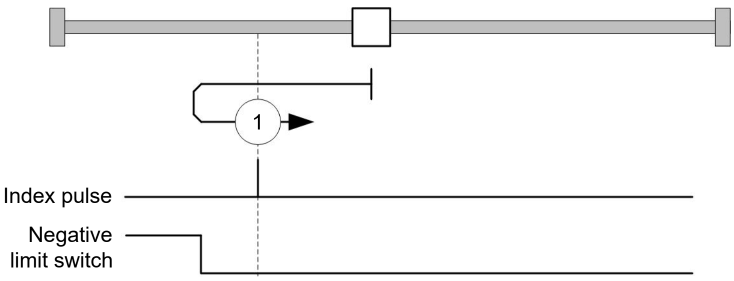
The following methods can be used for homing on block:

- Methods -1 to -2 and -7 to -14 contain an index pulse
- Methods -17 to -18 and -23 to -30 have no index pulse

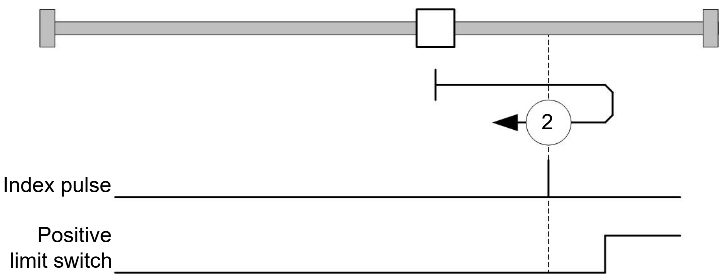
6.5.2.4 Methods 1 and 2

Reference to limit switches and index pulse.

Method 1 references to negative limit switch and index pulse:



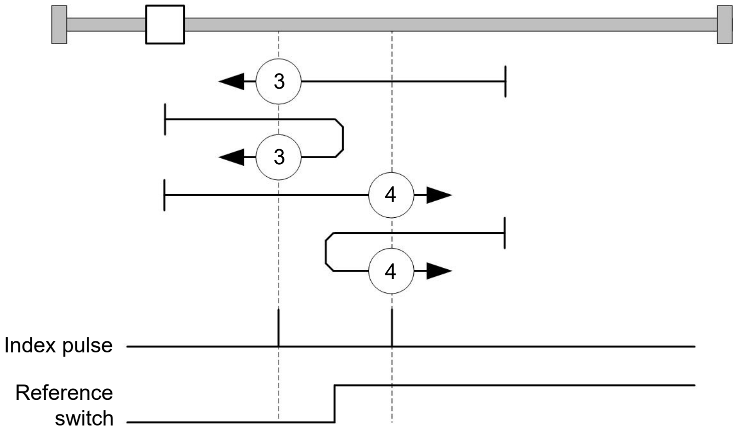
Method 2 references to positive limit switch and index pulse:



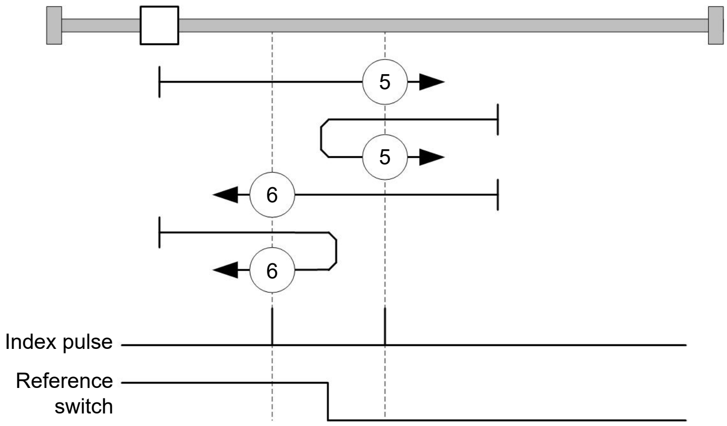
6.5.2.5 Methods 3 to 6

Reference to the switching edge of the home switch and index pulse.

With methods 3 and 4, the left switching edge of the home switch is used as reference:



With methods 5 and 6, the right switching edge of the home switch is used as reference:

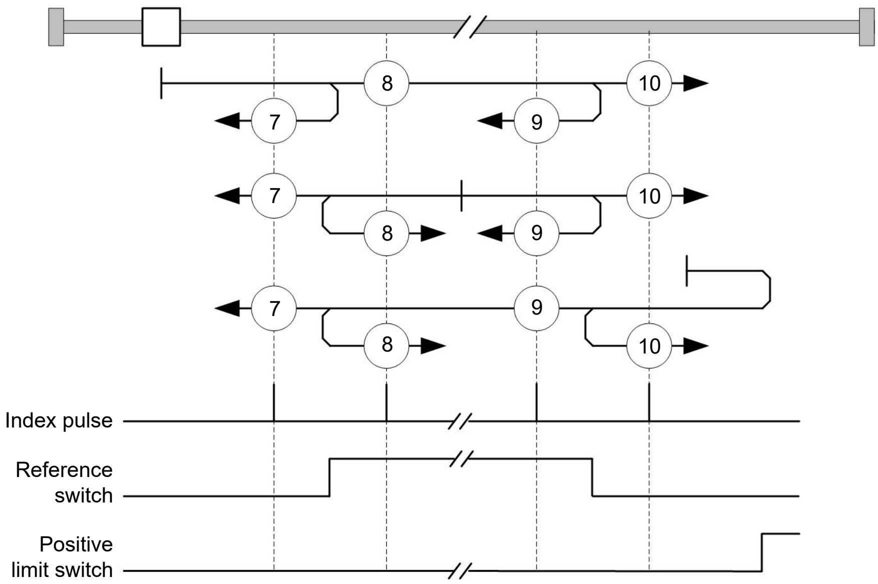


6.5.2.6 Methods 7 to 14

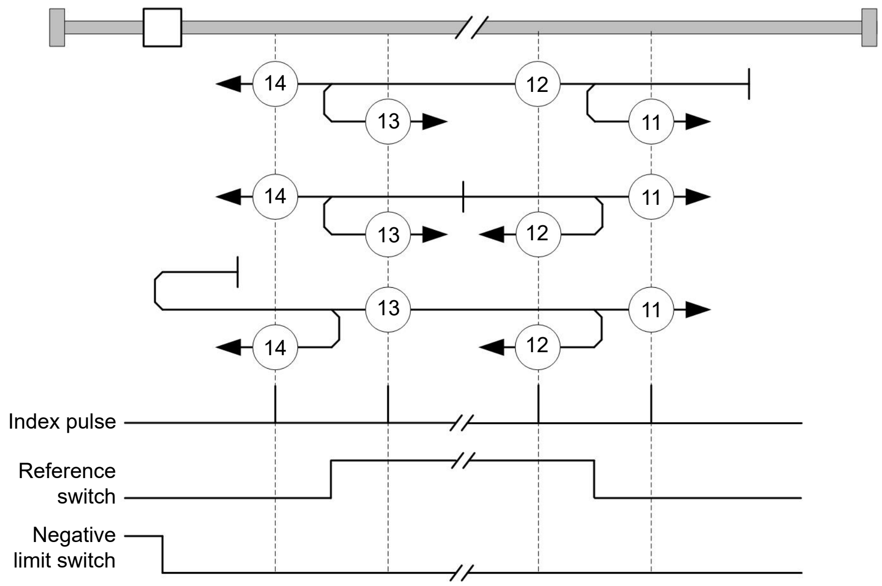
Reference to the home switch and index pulse (with limit switches).

With these methods, the current position relative to the home switch is not important. With method 10, for example, referencing is always performed to the index pulse to the right of the right edge of the home switch.

Methods 7 to 10 take the positive limit switch into account:



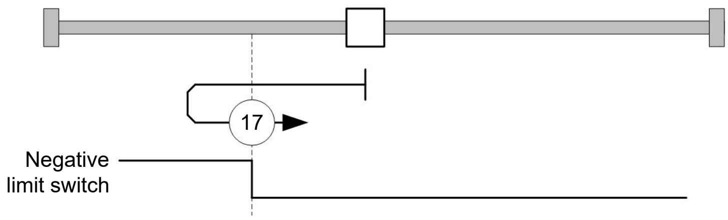
Methods 11 to 14 take the negative limit switch into account:



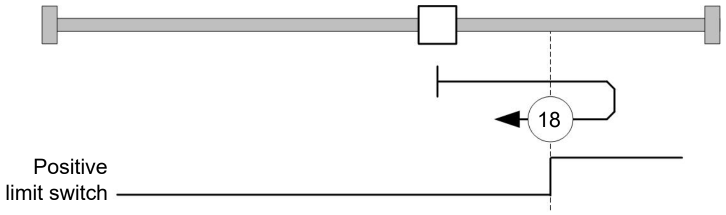
6.5.2.7 Methods 17 and 18

Reference to the limit switch without the index pulse.

Method 17 references to the negative limit switch:



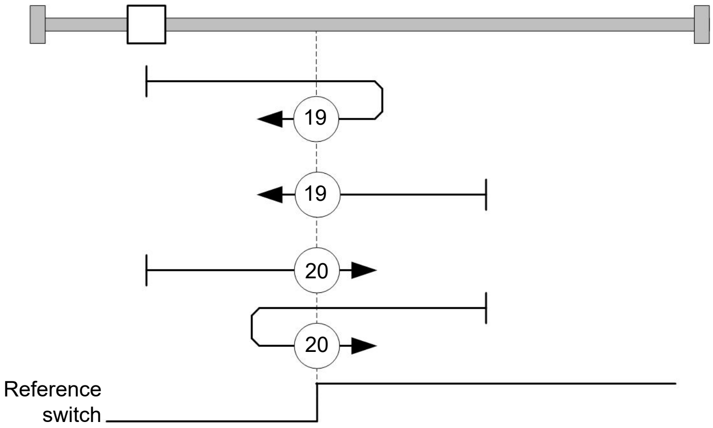
Method 18 references to the positive limit switch:



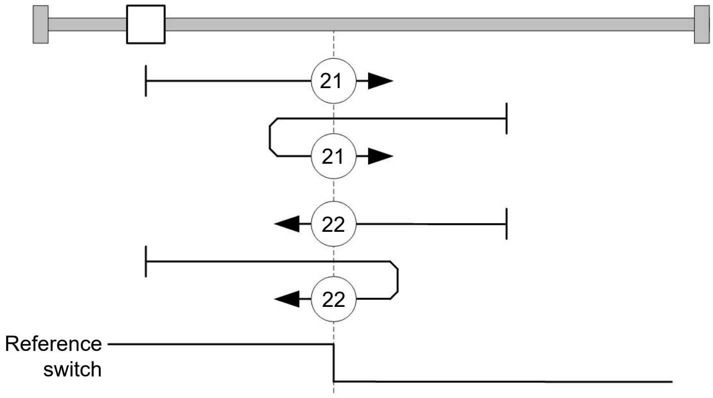
6.5.2.8 Methods 19 to 22

Reference to the switching edge of the home switch without the index pulse.

With methods 19 and 20 (equivalent to methods 3 and 4), the left switching edge of the home switch is used as reference:



With methods 21 and 22 (equivalent to methods 5 and 6), the right switching edge of the home switch is used as reference:

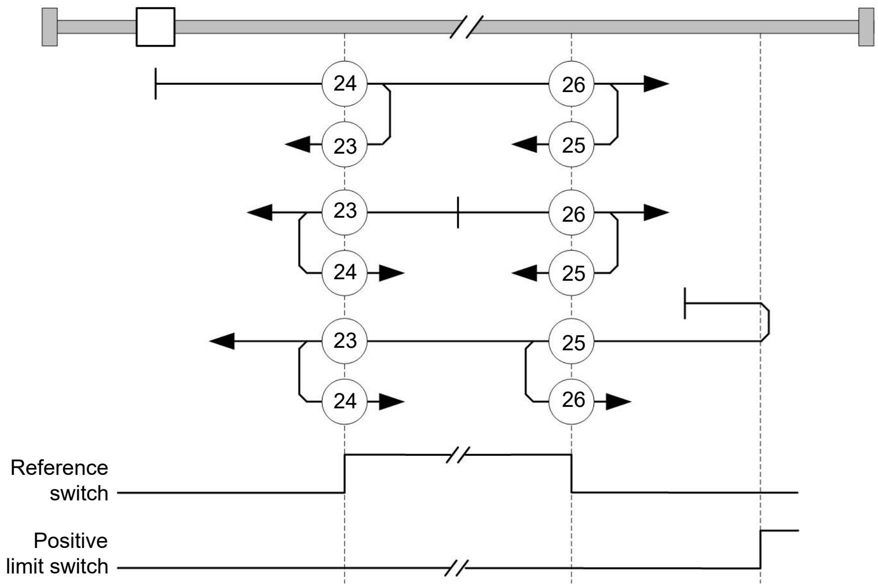


6.5.2.9 Methods 23 to 30

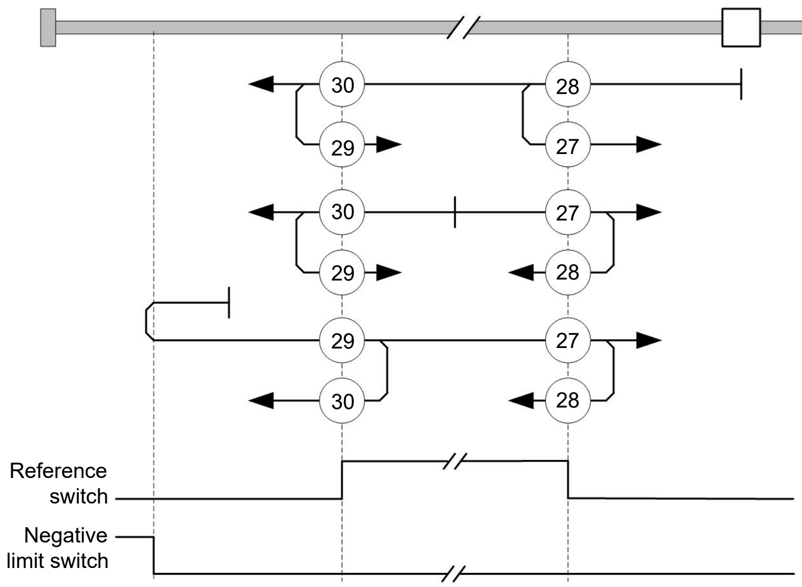
Reference to the home switch without the index pulse (with limit switches).

With these methods, the current position relative to the home switch is not important. With method 26, for example, referencing is always performed to the index pulse to the right of the right edge of the home switch.

Methods 23 to 26 take the positive home switch into account:



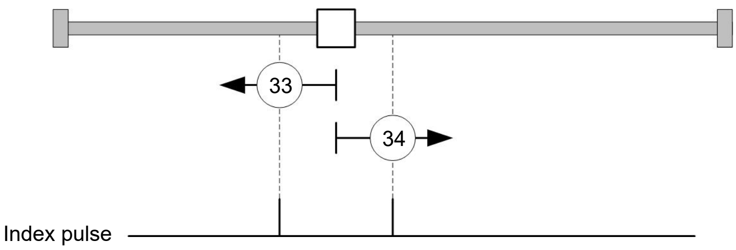
Methods 27 to 30 take the negative home switch into account:



6.5.2.10 Methods 33 and 34

Reference to the next index pulse.

With these methods referencing is only performed to the respective subsequent index pulse:



6.5.2.11 Method 35

References to the current position.

NOTICE

For homing mode 35, it is not necessary to switch the CiA 402 Power State Machine to the "Operation enabled" state. When energizing the motor windings in *open-loop* mode, it is thereby possible to prevent the current position from not being exactly 0 after Homing Mode 35.

6.6 Interpolated Position Mode

6.6.1 Overview

6.6.1.1 Description

Interpolated position mode is used to synchronize multiple axes. For this purpose, a higher-level controller performs the ramp and path calculation and passes the respective demand position, at which the axis is to be located at a certain time, to the controller. The controller interpolates between these intermediate position points.

6.6.1.2 Synchronization with the SYNC object

For interpolated position mode, it is necessary that the controller synchronizes with the SYNC object (depending on the fieldbus). This SYNC object is to be sent by the higher-level controller in regular intervals. Synchronization occurs as soon as the controller is switched to the *Operational* NMT mode.

NOTICE

Where possible, it is recommended that a time interval of the *SYNC object* be used.

6.6.2 Activation

To activate the mode, the value "7" must be set in object 6060_h (Modes Of Operation) (see "CiA 402 Power State Machine").

6.6.3 Controlword

The following bits in object 6040_h (controlword) have a special function:

- Bit 4 activates the interpolation when it is set to "1".
- Bit 8 (Halt): If this bit is set to "1", the motor stops. On a transition from "1" to "0", the motor accelerates with the set start ramp to the target speed. On a transition from "0" to "1", the motor brakes and comes to a standstill. The braking deceleration is dependent here on the setting of the "Halt Option Code" in object $605D_h$.

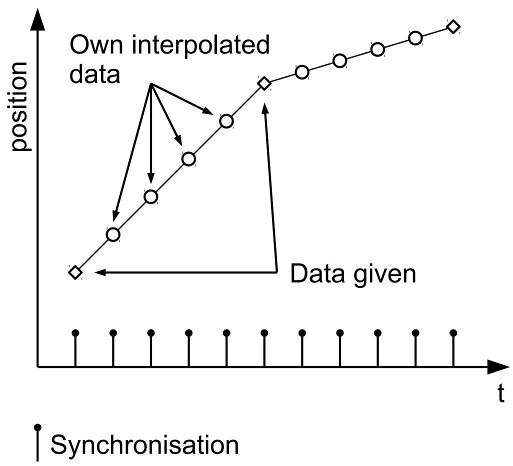
6.6.4 Statusword

The following bits in object 6041_h (statusword) have a special function:

- Bit 10: Target position reached: This bit is set to "1" if the target position was reached (if the halt bit in the controlword is "0") or the axis has speed 0 (if the halt bit in the last control word was "1").
- Bit 12 (IP mode active): This bit is set to "1" if interpolation is active.
- Bit 13 (Following Error): This bit is set in *closed loop* mode if the following error is greater than the set limits (6065_h (Following Error Window) and 6066_h (Following Error Time Out)).

6.6.5 Use

The controller follows a linearly interpolated path between the current position and the preset target position. The (next) target position must be written in record $60C1_h$:01_h.



In the current implementation, only

- linear interpolation
- and a target position

are supported.

6.6.6 Setup

The following setup is necessary:

- $60C2_h$:01_h: Time between two passed target positions in ms.

- 60C4$_h$:06$_h$: This object is to be set to "1" to be able to modify the target position in object 60C1$_h$:01$_h$.
- 6081$_h$ (Profile Velocity): Maximum speed with which the position is to be approached
- 6084$_h$ (Profile Deceleration): Desired braking deceleration during braking
- 60C6$_h$: (Max Deceleration): The maximum allowed braking deceleration
- To be able to turn the motor, the *power state machine* is to be set to the *Operation enabled* state (see CiA 402 Power State Machine).

6.6.7 Operation

After setting up, the task of the higher-level controller is to write the target positions to object 60C1$_h$:01$_h$ in time.

6.7 Cyclic Synchronous Position

6.7.1 Overview

6.7.1.1 Description

In this mode, the controller receives an absolute position preset via the fieldbus at fixed time intervals (referred to in the following as a *cycle*). The controller then no longer calculates any ramps, but rather only follows the presets.

The target position is transferred cyclically (via *PDO*). Bit 4 in the controlword does not need to be set (unlike the Profile Position mode).

NOTICE

The target is absolute and, thus, independent of how often it was sent per *cycle*.

6.7.1.2 Synchronization with the SYNC object

To achieve smooth movement, the controller should synchronize with the SYNC object (depending on the field bus). This SYNC object is to be sent by the higher-level controller in regular intervals. Synchronization occurs as soon as the controller is switched to the *Operational* NMT mode.

NOTICE

Where possible, it is recommended that a time interval of the *SYNC object* be used for transfer of the target position.

6.7.1.3 Activation

To activate the mode, the value "8" must be set in object 6060$_h$ (Modes Of Operation) (see "CiA 402 Power State Machine").

6.7.1.4 Controlword

In this mode, the bits of controlword 6040$_h$ have no special function.

6.7.1.5 Statusword

The following bits in object 6041$_h$ (statusword) have a special function:

Bit	Value	Description
8	0	The controller is not in sync with the fieldbus
8	1	The controller is in sync with the fieldbus
10	0	Reserved

Bit	Value	Description
10	1	Reserved
12	0	Controller does not follow the target; the preset of 607A$_h$ (Target Position) is ignored
12	1	Controller follows the target; object 607A$_h$ (Target Position) is used as the input for position control.
13	0	No following error
13	1	Following error

Bit 11: Limit exceeded: The demand position is above or below the limit values set in 607D$_h$.

6.7.2 Object entries

The following objects are necessary for controlling this mode:

- 607A$_h$ (Target Position): This object must be written cyclically with the position set value.
- 607B$_h$ (Position Range Limit): This object contains the preset for an overrun or underrun of the position specification.
- 607D$_h$ (Software Position Limit): This object defines the limitations within which the position specification (607A$_h$) must be located.
- 6065$_h$ (Following Error Window): This object specifies a tolerance corridor in both the positive and negative direction from the set specification. If the actual position is outside of this corridor for longer than the specified time (6066$_h$), a following error is reported.
- 6066$_h$ (Following Error Time Out): This object specifies the time range in milliseconds. If the actual position is outside of the position corridor (6065$_h$) for longer than this time range, a following error is triggered.
- 6085$_h$ (Quick-Stop Deceleration): This object contains the braking deceleration for the case that a quick-stop is triggered.
- 605A$_h$ (Quick-Stop Option Code): This object contains the option that is to be executed in the event of a quick-stop.
- 60C2$_h$:01$_h$ (Interpolation Time Period): This object specifies the time of a *cycle*; a new set value must be written in 607A$_h$ in these time intervals.
  The following applies here: cycle time = value of 60C2$_h$:01$_h$ * $10^{\text{value of 60C2:02}}$ seconds.
- 60C2$_h$:02$_h$ (Interpolation Time Index): This object specifies the time basis of the cycles. Currently, only value 60C2$_h$:02$_h$=-3 is supported; this yields a time basis of 1 millisecond.
- 60B0$_h$ (Position Offset): Offset for the position set value in user-defined units
- 60B1$_h$ (Velocity Offset): Offset for the speed set value in user-defined units
- 60B2$_h$ (Torque Offset): Offset for the torque set value in tenths of a percent

The following objects can be read in this mode:

- 6064$_h$ (Position Actual Value)
- 606C$_h$ (Velocity Actual Value)
- 60F4$_h$ (Following Error Actual Value)

6.8 Cyclic Synchronous Velocity

6.8.1 Overview

6.8.1.1 Description

In this mode, the controller passes a speed preset via the fieldbus at fixed time intervals (referred to in the following as a *cycle*). The controller then no longer calculates any ramps, but rather only follows the presets.

6.8.1.2 Activation

To activate the mode, the value "9" must be set in object 6060_h (Modes Of Operation) (see "CiA 402 Power State Machine").

6.8.1.3 Controlword

In this mode, the bits of controlword 6040_h have no special function.

6.8.1.4 Statusword

The following bits in object 6041_h (statusword) have a special function:

Bit	Value	Description
8	0	The controller is not in sync with the fieldbus
8	1	The controller is in sync with the fieldbus
10	0	Reserved
10	1	Reserved
12	0	Controller does not follow the target; the preset of $60FF_h$ (Target Velocity) is ignored
12	1	Controller follows the target; object $60FF_h$ (Target Velocity) is used as the input for position control.
13	0	Reserved
13	1	Reserved

6.8.2 Object entries

The following objects are necessary for controlling this mode:

- $60FF_h$ (Target Velocity): This object must be written cyclically with the speed set value.
- 6085_h (Quick-Stop Deceleration): This object contains the braking deceleration for the case that a quick-stop is triggered (see "CiA 402 Power State Machine").
- $605A_h$ (Quick-Stop Option Code): This object contains the option that is to be executed in the event of a quick-stop (see "CiA 402 Power State Machine").
- 6080_h (Max Motor Speed): Maximum speed
- $60C2_h$:01_h (Interpolation Time Period): This object specifies the time of a *cycle*; a new set value must be written in $60FF_h$ in these time intervals.
  The following applies here: cycle time = value of $60C2_h$:01_h * $10^{\text{value of 60C2:02}}$ seconds.
- $60C2_h$:02_h (Interpolation Time Index): This object specifies the time basis of the cycles. Currently, only value $60C2_h$:02_h=-3 is supported; this yields a time basis of 1 millisecond.
- $60B1_h$ (Velocity Offset): Offset for the speed set value in user-defined units
- $60B2_h$ (Torque Offset): Offset for the torque set value in tenths of a percent

The following objects can be read in this mode:

- $606C_h$ (Velocity Actual Value)
- $607E_h$ (Polarity)

6.9 Cyclic Synchronous Torque

6.9.1 Overview

6.9.1.1 Description

In this mode, the controller passes an absolute torque preset via the fieldbus at fixed time intervals (referred to in the following as a *cycle*). The controller then no longer calculates any ramps, but rather only follows the presets.

NOTICE

This mode only functions if closed loop is activated, see also Commissioning closed loop.

6.9.1.2 Activation

To activate the mode, the value "10" must be set in object 6060_h (Modes Of Operation) (see "CiA 402 Power State Machine").

6.9.1.3 Controlword

In this mode, the bits of controlword 6040_h have no special function.

6.9.1.4 Statusword

The following bits in object 6041_h (statusword) have a special function:

Bit	Value	Description
8	0	The controller is not in sync with the fieldbus
8	1	The controller is in sync with the fieldbus
10	0	Reserved
10	1	Reserved
12	0	Controller does not follow the target; the preset of 6071_h (Target Torque) is ignored
12	1	Controller follows the target; object 6071_h (Target Torque) is used as the input for position control.
13	0	Reserved
13	1	Reserved

6.9.2 Object entries

The following objects are necessary for controlling this mode:

- 6071_h (Target Torque): This object must be written cyclically with the torque set value and is to be set relative to 6072_h.
- 6072_h (Max Torque): Describes the maximum permissible torque.
- 6073_h (Max Current):
  Maximum current. The minimum of 6073_h and 6072_h is used as limit for the torque in 6071_h.
- 6080_h (Max Motor Speed): Maximum speed
- $60C2_h$:01_h (Interpolation Time Period): This object specifies the time of a *cycle*; a new set value must be written in 6071_h in these time intervals.
  The following applies here: cycle time = value of $60C2_h$:01_h * $10^{\text{value of 60C2:02}}$ seconds.
- $60C2_h$:02_h (Interpolation Time Index): This object specifies the time basis of the cycles. Currently, only value $60C2_h$:02_h=-3 is supported; this yields a time basis of 1 millisecond.
- $60B2_h$ (Torque Offset): Offset for the torque set value in tenths of a percent

The following objects can be read in this mode:

- $606C_h$ (Velocity Actual Value)
- 6074_h (Torque Demand)

6.10 Clock-direction mode

6.10.1 Description

In clock-direction mode, the motor is operated via two inputs by a higher-level positioning controller with clock and direction signal. On each clock signal, the motor moves one step in the direction corresponding to the direction signal.

NOTICE

This mode is not available with product variants PD1-…-65-….

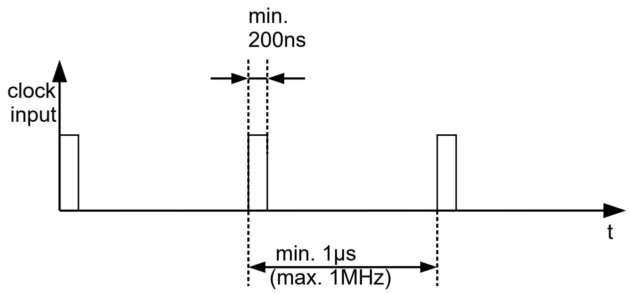
6.10.2 Activation

To activate the mode, the value "-1" (or "FF_h") must be set in object 6060_h (Modes Of Operation) (see "CiA 402 Power State Machine").

6.10.3 General

The following data apply for every subtype of the clock-direction mode:

- The maximum frequency of the input pulse is 1 MHz; the ON pulse should not be less than 200 ns.



- The demand position resulting from the input pulses is updated cyclically; the cycle time corresponds to the Interpolation Time Period ($60C2_h$). The input pulses that arrive within a cycle are collected and buffered in the controller.
- The steps are scaled using objects 2057_h and 2058_h. The following formula applies here:

$$\text{step width per pulse} = \frac{2057_h}{2058_h}$$

The "step size per pulse" value is set to 128 (2057_h=128 and 2058_h=1) ex works, which corresponds to a quarter step per pulse. A full step is the value "512", a half step per pulse corresponds to "256", etc.

NOTICE

For a stepper motor with 50 pole pairs, 200 full steps correspond to one mechanical revolution of the motor shaft.
In *clock-direction mode*, the BLDC motors are also handled as stepper motors by the controller. This means that for a BLDC motor with, e.g., 3 pole pairs, 12 (=4*3) full steps correspond to one revolution.

NOTICE

If there is a change of direction, a time of at least 35 µs must elapse before the new clock signal is applied.

6.10.4 Statusword

The following bits in object 6041$_h$ (statusword) have a special function:

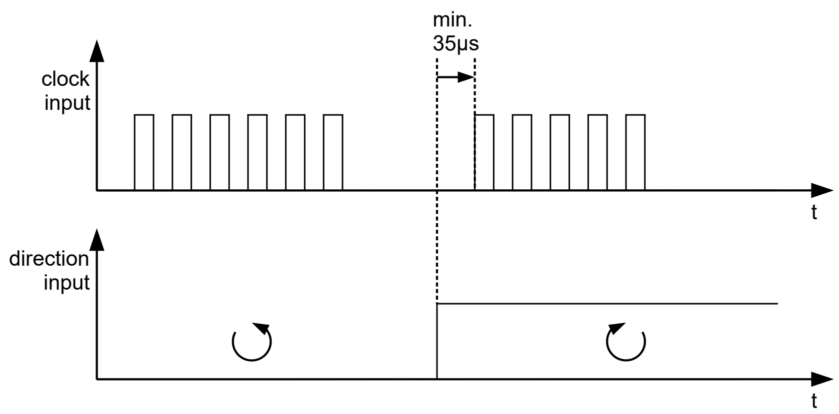
- Bit 13 (Following Error): This bit is set in *closed loop* mode if the following error is greater than the set limits (6065$_h$ (Following Error Window) and 6066$_h$ (Following Error Time Out)).

6.10.5 Subtypes of the clock-direction mode

6.10.5.1 Clock-direction mode (TR mode)

To activate the mode, object 205B$_h$ must be set to the value "0" (factory settings).

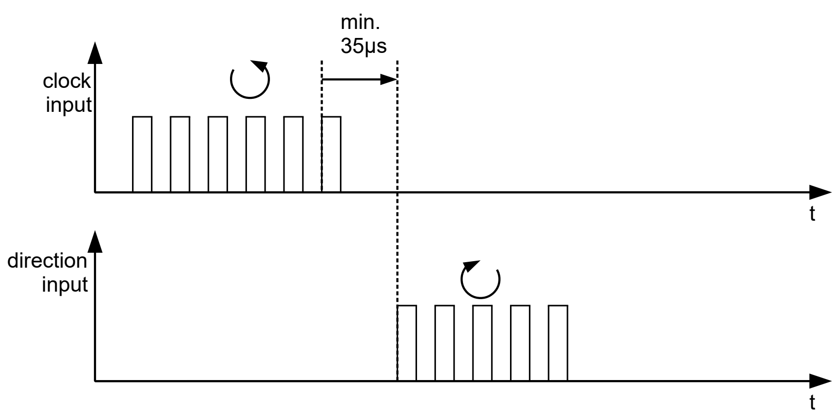
In this mode, the pulses must be preset via the clock input; the signal of the direction input specifies the direction of rotation here (see following graphic).



6.10.5.2 Right / left rotation mode (CW / CCW mode)

To activate the mode, object 205B$_h$ must be set to the value "1".

In this mode, the input that is used decides the direction of rotation (see following graphic).



6.11 Auto setup

6.11.1 Description

To determine a number of parameters related to the motor and the connected sensors (encoders/Hall sensors), an *auto setup* is performed. Closed-Loop operation requires a successfully completed *auto setup*. For details, see the corresponding section in chapter Commissioning.

6.11.2 Activation

To activate the mode, the value "-2" (="FE_h") must be set in object 6060_h (Modes Of Operation) (see CiA 402 Power State Machine).

6.11.3 Controlword

The following bits in object 6040_h (controlword) have a special function:

- Bit 4 starts a travel command. This is carried out on a transition from "0" to "1".

6.11.4 Statusword

The following bits in object 6041_h (statusword) have a special function:

- Bit 10: Indexed: indicates whether (= "1") or not (= "0") an encoder index was found.
- Bit 12: Aligned: this bit is set to "1" after *auto setup* has concluded

7 Special functions

7.1 Digital inputs and outputs

This product is equipped with digital inputs and outputs. You can find the exact number for the given product variant in chapter Pin assignment.

In 323Ah User Pin Settings, you configure the hardware as follows:

- Subindex 01_h: Here, you define the level for the inputs/outputs:
  - Value "0": 5 V
  - Value "1": 24 V (inputs) or +UB (outputs)

NOTICE

Use for the inputs a voltage that is smaller than the operating voltage +UB.

- Subindex 02_h: Here, you define the wiring for the digital inputs:
  - Value "0" (Pull-Down): High level when 5/24 V at Pin.
  - Value "1" (Pull-Up): High level without external voltage at Pin.

7.1.1 Digital inputs

7.1.1.1 Overview

NOTICE

For digital inputs with 5 V, the length of the supply lines must not exceed 3 meters.

NOTICE

The digital inputs are sampled once per millisecond. Signal changes at the input less than one millisecond in duration are not processed.

7.1.1.2 Computation of the inputs

Object $60FD_h$ (Digital Inputs) contains a summary of the inputs and the special functions. The current status of the inputs is likewise read out from object 324Ah Inputs (including Hall sensors and incremental encoders, if present).

The following table lists the value of the corresponding bit in the respective object for the inputs depending on the configuration in 323Ah User Pin Settings:

Voltage at pin	Subindex 02 (Pull-Up Enable)	Subindex 01 (Voltage Level Select)	Bit value
n.c	0 (Pull-Down)	X	0
GND	0 (Pull-Down)	X	0
5 V	0 (Pull-Down)	0 (5 V)	1
5 V	0 (Pull-Down)	1 (24 V)	0
24 V	0 (Pull-Down)	1 (24 V)	1
n.c.	1 (Pull-Up)	X	1
GND	1 (Pull-Up)	X	0

Voltage at pin	Subindex 02 (Pull-Up Enable)	Subindex 01 (Voltage Level Select)	Bit value
5 V	1 (Pull-Up)	0 (5 V)	1
5 V	1 (Pull-Up)	1 (24 V)	0
24 V	1 (Pull-Up)	1 (24 V)	1

7.1.1.3 Special functions

The firmware evaluates the following bits in $60FD_h$:

- Bit 0: Negative limit switch (see Limitation of the range of motion)
- Bit 1: Positive limit switch (see Limitation of the range of motion)
- Bit 2: Home switch (see Homing)
- Bit 3: Interlock (see interlock function)

You define the assignment of the bits to the pins with the *Input Routing*.

7.1.1.4 Input Routing

Principle

To perform the assignment of the inputs more flexibly, there is a mode called *Input Routing Mode*. This assigns a signal of a source to a bit in object $60FD_h$.

Routing

Object 3242_h determines which signal source is routed to which bit of $60FD_h$. Subindex 01_h of 3242_h determines bit 0, subindex 02_h determines bit 1, and so forth. The signal sources and their numbers can be found in the following lists.

Number		
dec	hex	Signal source
00	00	Signal is always 0
01	01	physical input 1
02	02	Physical input 2
03	03	Physical input 3
04	04	Physical input 4
05	05	Physical input 5
06	06	Physical input 6
07	07	Physical input 7
08	08	Physical input 8
09	09	Physical input 9
10	0A	physical input 10
11	0B	Physical input 11
12	0C	physical input 12
13	0D	Physical input 13
14	0E	Physical input 14

Number		
dec	**hex**	**Signal source**
15	0F	Physical input 15
16	10	Physical input 16
65	41	Hall input "U"
66	42	Hall input "V"
67	43	Hall input "W"
68	44	Encoder input "A"
69	45	Encoder input "B"
70	46	Encoder input "Index"
81	51	Negative block
82	52	Positive block
90	5A	Analog input

The following table describes the inverted signals of the previous table.

Number		
dec	**hex**	**Signal source**
128	80	Signal is always 1
129	81	Inverted physical input 1
130	82	Inverted physical input 2
131	83	Inverted physical input 3
132	84	Inverted physical input 4
133	85	Inverted physical input 5
134	86	Inverted physical input 6
135	87	Inverted physical input 7
136	88	Inverted physical input 8
137	89	Inverted physical input 9
138	8A	Inverted physical input 10
139	8B	Inverted physical input 11
140	8C	Inverted physical input 12
141	8D	Inverted physical input 13
142	8E	Inverted physical input 14
143	8F	Inverted physical input 15
144	90	Inverted physical input 16
193	C1	Inverted Hall input "U"
194	C2	Inverted Hall input "V"
195	C3	Inverted Hall input "W"
196	C4	Inverted encoder input "A"
197	C5	Inverted encoder input "B"
198	C6	Inverted encoder input "Index"

Example

Input 1 is to be routed to bit 16 of object $60FD_h$:

The number of the signal source for input 1 is "1". The routing for bit 16 is written in 3242_h:11_h.

Hence, object 3242_h:11_h must be set to the value "1".

7.1.1.5 Interlock function

The interlock function is a release that you control via bit 3 in $60FD_h$. If this bit is set to "1", the motor can move. If the bit is set to "0", the controller switches to the error state and the action stored in $605E_h$ is executed.

To activate the interlock function, you must switch on the special function by setting bit 3 in 3240:01_h to "1".

Use *Input Routing* to define which signal source is routed to bit 3 of $60FD_h$ and is to control the interlock function.

Example

Input 4 is to be routed to bit 3 of object $60FD_h$ to control the interlock function. A low level is to result in an error state.

1. To route input 4 to bit 3, set 3242_h:04_h to "4".

7.1.2 Digital outputs

7.1.2.1 Outputs

The outputs are controlled via object $60FE_h$. Here, output 1 corresponds to bit 16 in object $60FE_h$, output 2 corresponds to bit 17, etc., as with the inputs. The outputs with special functions are entered in the firmware in the lower bits 0 to 15. The only bit assigned at the present time is bit 0, which controls the motor brake.

7.1.2.2 Wiring

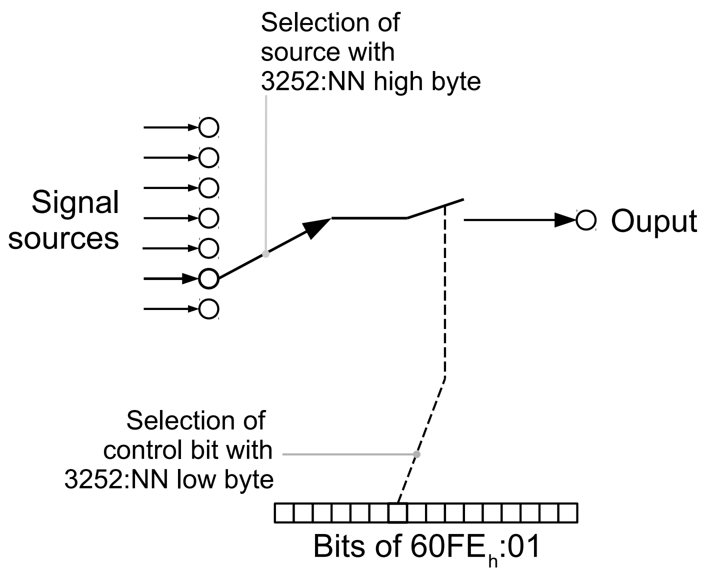
The digital outputs are push-pull. The voltage at the pin in the high state is either 5 V ($323A_h$:01_h=0) or +Up ($323A_h$:01_h=1). The current should not exceed 50 mA.

7.1.2.3 Output Routing

Principle

The "Output Routing Mode" assigns an output a signal source; a control bit in object $60FE_h$:01_h switches the signal on or off.

The source is selected with 3252_h:01 to n in the "high byte" (bit 15 to bit 8). The assignment of a control bit from object $60FE_h$:01_h is performed in the "low byte" (bit 7 to bit 0) of 3252_h:01_h to n (see following figure).



Routing

The subindex of object 3252_h determines which signal source is routed to which output. The output assignments are listed in the following:

NOTICE

The maximum output frequency of the PWM output (software PWM) is 2 kHz. All other outputs can only produce signals up to 500 Hz.

Subindices 3252_h:01_h to $0n_h$ are 16 bits wide, whereby the high byte selects the signal source (e. g., the PWM generator) and the low byte determines the control bit in object $60FE_h$:01.

Bit 7 of 3252_h:01_h to $0n_h$ inverts the controller from object $60FE_h$:01. Normally, value "1" in object $60FE_h$:01_h switches on the signal; if bit 7 is set, the value "0" switches on the signal.

Number in 3252:01 to 0n	
$00XX_h$	Output is always "1"
$01XX_h$	Output is always "0"
$02XX_h$	Encoder signal (6063_h) with frequency divider 1
$03XX_h$	Encoder signal (6063_h) with frequency divider 2
$04XX_h$	Encoder signal (6063_h) with frequency divider 4
$05XX_h$	Encoder signal (6063_h) with frequency divider 8
$06XX_h$	Encoder signal (6063_h) with frequency divider 16
$07XX_h$	Encoder signal (6063_h) with frequency divider 32
$08XX_h$	Encoder signal (6063_h) with frequency divider 64
$09XX_h$	Position Actual Value (6064_h) with frequency divider 1
$0AXX_h$	Position Actual Value (6064_h) with frequency divider 2
$0BXX_h$	Position Actual Value (6064_h) with frequency divider 4
$0CXX_h$	Position Actual Value (6064_h) with frequency divider 8
$0DXX_h$	Position Actual Value (6064_h) with frequency divider 16
$0EXX_h$	Position Actual Value (6064_h) with frequency divider 32
$0FXX_h$	Position Actual Value (6064_h) with frequency divider 64

NOTICE

On any change of the "encoder signal" (6063_h) or the current position (6064_h in user-defined units) by an increment, a pulse is output at the digital input (for frequency divider 1). Take this into account when selecting the frequency divider and the unit, especially when using sensors with low resolution (such as Hall sensors).

Example

The encoder signal (6063_h) is to be applied to output 1 with a frequency divider 4. The output is to be controlled with bit 5 of object 60FE:01.

- 3250_h:08_h = 1 (activate routing)
- 3252_h:02_h = 0405_h ($04XX_h$ + 0005_h)
- $04XX_h$: Encoder signal with frequency divider 4
- 0005_h: Selection of bit 5 of 60FE:01

The output is switched on by setting bit 5 in object 60FE:01.

Example

The brake PWM signal is to be applied to output 2. Because the automatic brake control uses bit 0 of 60FE:01_h, this should be used as control bit.

- 3250_h:08_h = 1 (activate routing)
- 3252_h:03_h = 1080_h (=$10XX_h$ + 0080_h). Where:
  - $10XX_h$: Brake PWM signal
  - 0080_h: Selection of the inverted bit 0 of object 60FE:01

7.2 I^2t Motor overload protection

7.2.1 Description

NOTICE

For stepper motors, only the rated current is specified, not a maximum current. No liability is therefore assumed when using I^2t with stepper motors.

The goal of I^2t motor overload protection is to protect the motor from damage and, at the same time, operate it normally up to its thermal limit.

This function is only available if the controller is in the closed loop mode (bit 0 of object 3202_h must be set to "1").

7.2.2 Object entries

The following objects affect I^2t motor overload protection:

- 2031_h: Max Motor Current – specifies the maximum permissible motor current in mA.
- $203B_h$:1_h Motor Rated Current – specifies the rated current in mA.
- 6073_h Max Current – specifies the maximum current in tenths of a percent of the set rated current.
- $203B_h$:2_h Maximum Duration Of Peak Current – specifies the maximum duration of the maximum current in ms.

The following objects indicate the current state of I^2t:

- 203B$_h$:3$_h$ Threshold – specifies the limit in A^2ms that determines whether the maximum current or rated current is switched to.
- 203B$_h$:4$_h$ CalcValue – specifies the calculated value in A^2ms that is compared with the threshold for setting the current.
- 203B$_h$:5$_h$ LimitedCurrent – shows the momentary current value in mA that was set by I^2t.
- 203B$_h$:6$_h$ Status:
  - Value = "0": I^2t deactivated
  - Value = "1": I^2t activated

7.2.3 Activation

Closed loop must be activated, (bit 0 of object 3202$_h$ set to "1", see also chapter Closed-Loop).

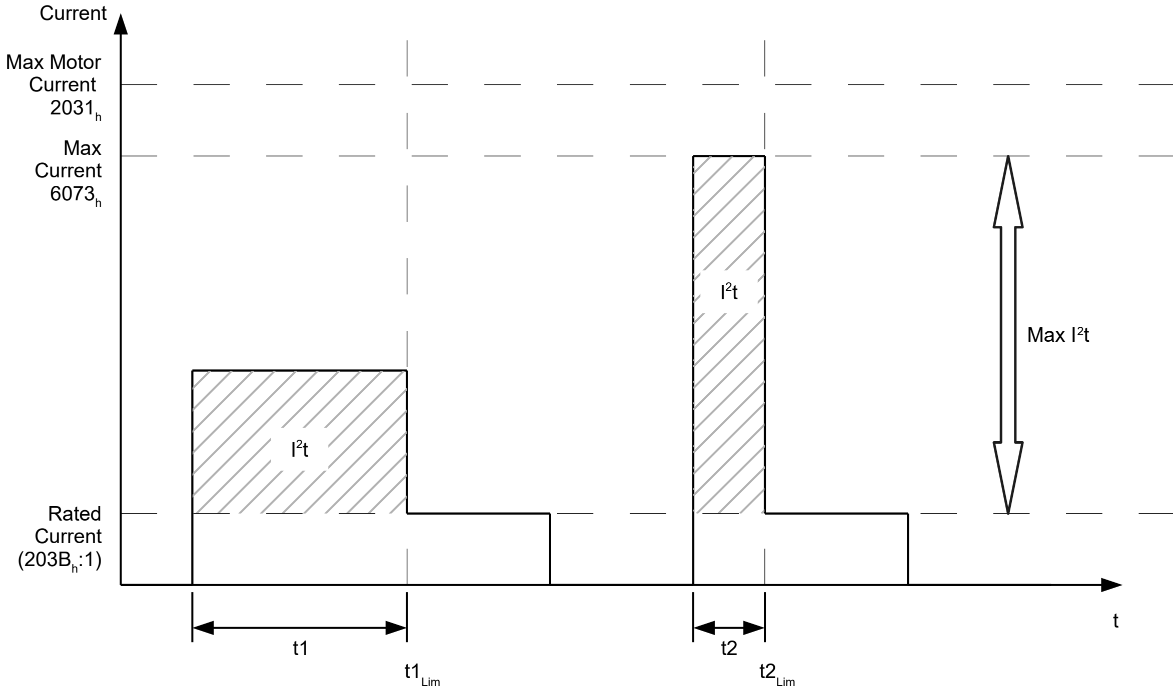
To activate the mode, you must appropriately specify the four object entries mentioned above (2031$_h$, 6073$_h$, 203B$_h$:1$_h$, 203B$_h$:2$_h$). This means that the maximum current must be greater than the rated current and a time value for the maximum duration of the maximum current must be entered. If these conditions are not met, the I^2t functionality remains deactivated.

7.2.4 Function of I^2t

From the specification of rated current, maximum current and maximum duration of the maximum current, an I^2t_{Lim} is calculated.

The motor can run with maximum current until the calculated I^2t_{Lim} is reached. The current is then immediately reduced to the rated current. The maximum current is limited by the maximum motor current (2031$_h$).

The relationships are illustrated again in the following diagrams.



In the first section, t1, the current value is higher than the rated current. At time t1$_{Lim}$, I^2t_{Lim} is reached and the current is limited to the rated current. A current that corresponds to the maximum current then occurs for a period of time t2. Hence, the value for I^2t_{Lim} is reached more quickly than in time t1.

7.3 Saving objects

NOTICE

Improper use of the function can result in it no longer being possible to start the controller. Therefore, carefully read the entire chapter before using the function.

7.3.1 General

Many objects in the object dictionary can be saved and then automatically reloaded the next time the controller is switched on or reset. Furthermore, the saved values are also retained following a firmware update.

Only entire collections of objects (referred to in the following as *categories*) can be saved together; individual objects cannot be saved.

An object can be assigned one of the following *categories*:

- Communication: Parameters related to external interfaces, such as PDO configuration etc.
- Application: Parameters related to operating modes.
- User: Parameters that are written and read by the customer/user only and are ignored by the controller firmware.
- Drive: Parameters related to the motor and the sensors (BLDC/Stepper, *closed/open-loop*…). Some are set and saved by auto setup.
- Tuning: Parameters related to motor and encoder that are set either by auto setup or that can be found in the data sheets, e.g., pole pairs and maximum current.
- Modbus RTU: Parameters related to Modbus RTU communication

If an object is not assigned one of these *categories*, it cannot be saved, e.g., statusword and all objects whose value is dependent on the current state of the controller.

The objects in each *category* are listed below. In chapter Description of the object dictionary, the corresponding *category* for each object is also specified.

7.3.2 Category: Communication

- 1005_h: COB-ID Sync
- 1006_h: Communication Cycle Period
- 1007_h: Synchronous Window Length
- 1014_h: COB-ID EMCY
- 2102_h: Fieldbus Module Control
- 3502_h: MODBUS Rx PDO Mapping
- 3602_h: MODBUS Tx PDO Mapping

7.3.3 Category: Application

- 2034_h: Upper Voltage Warning Level
- 2035_h: Lower Voltage Warning Level
- 2036_h: Open Loop Current Reduction Idle Time
- 2037_h: Open Loop Current Reduction Value/factor
- 2038_h: Brake Controller Timing
- $203A_h$: Homing On Block Configuration
- $203D_h$: Torque Window
- $203E_h$: Torque Window Time Out
- $203F_h$: Max Slippage Time Out
- 2057_h: Clock Direction Multiplier
- 2058_h: Clock Direction Divider
- $205B_h$: Clock Direction Or Clockwise/Counter Clockwise Mode

- 2084_h: Bootup Delay
- 2290_h: PDI Control
- 2300_h: NanoJ Control
- 2410_h: NanoJ Init Parameters
- 2800_h: Bootloader And Reboot Settings
- 3212_h: Motor Drive Flags
- $321A_h$: Current Controller Parameters
- $321B_h$: Velocity Controller Parameters
- $321C_h$: Position Controller Parameters
- $321D_h$: Feedforward
- $321E_h$: Voltage Limit
- $323A_h$: User Pin Settings
- 3241_h: Digital Input Position Capture
- 3242_h: Digital Input Routing
- 3243_h: Home Switch Position Capture
- 3250_h: Digital Outputs Control
- 3252_h: Digital Output Routing
- $325A_h$: Outputs
- 3273_h: Generic SPI Hardware Configuration
- 3274_h: Generic SPI Mosi Data
- 3321_h: Analog Input Offsets
- 3322_h: Analog Input Numerators
- 3323_h: Analog Input Denominators
- 3700_h: Deviation Error Option Code
- 3701_h: Limit Switch Error Option Code
- 4013_h: HW Configuration
- 4015_h: Special Drive Modes
- 6007_h: Abort Connection Option Code
- 6040_h: Controlword
- 6042_h: Vl Target Velocity
- 6046_h: Vl Velocity Min Max Amount
- 6048_h: Vl Velocity Acceleration
- 6049_h: Vl Velocity Deceleration
- $604A_h$: Vl Velocity Quick Stop
- $604C_h$: Vl Dimension Factor
- $605A_h$: Quick Stop Option Code
- $605B_h$: Shutdown Option Code
- $605C_h$: Disable Operation Option Code
- $605D_h$: Halt Option Code
- $605E_h$: Fault Reaction Option Code
- 6060_h: Modes Of Operation
- 6065_h: Following Error Window
- 6066_h: Following Error Time Out
- 6067_h: Position Window
- 6068_h: Position Window Time
- $606D_h$: Velocity Window
- $606E_h$: Velocity Window Time
- $606F_h$: Velocity Threshold
- 6070_h: Velocity Threshold Time
- 6071_h: Target Torque
- 6072_h: Max Torque
- $607A_h$: Target Position
- $607B_h$: Position Range Limit

- 607C$_h$: Home Offset
- 607D$_h$: Software Position Limit
- 607E$_h$: Polarity
- 607F$_h$: Max Profile Velocity
- 6081$_h$: Profile Velocity
- 6082$_h$: End Velocity
- 6083$_h$: Profile Acceleration
- 6084$_h$: Profile Deceleration
- 6085$_h$: Quick Stop Deceleration
- 6086$_h$: Motion Profile Type
- 6087$_h$: Torque Slope
- 6091$_h$: Gear Ratio
- 6092$_h$: Feed Constant
- 6096$_h$: Velocity Factor
- 6097$_h$: Acceleration Factor
- 6098$_h$: Homing Method
- 6099$_h$: Homing Speed
- 609A$_h$: Homing Acceleration
- 60A2$_h$: Jerk Factor
- 60A4$_h$: Profile Jerk
- 60A8$_h$: SI Unit Position
- 60A9$_h$: SI Unit Velocity
- 60B0$_h$: Position Offset
- 60B1$_h$: Velocity Offset
- 60B2$_h$: Torque Offset
- 60C1$_h$: Interpolation Data Record
- 60C2$_h$: Interpolation Time Period
- 60C4$_h$: Interpolation Data Configuration
- 60C5$_h$: Max Acceleration
- 60C6$_h$: Max Deceleration
- 60E8$_h$: Additional Gear Ratio - Motor Shaft Revolutions
- 60E9$_h$: Additional Feed Constant - Feed
- 60ED$_h$: Additional Gear Ratio - Driving Shaft Revolutions
- 60EE$_h$: Additional Feed Constant - Driving Shaft Revolutions
- 60F2$_h$: Position Option Code
- 60F8$_h$: Max Slippage
- 60FE$_h$: Digital Outputs
- 60FF$_h$: Target Velocity

7.3.4 Category: User

- 2701$_h$: Customer Storage Area

7.3.5 Category: Movement

- 3202$_h$: Motor Drive Submode Select
- 320D$_h$: Torque Of Inertia Factor
- 6073$_h$: Max Current
- 6080$_h$: Max Motor Speed

7.3.6 Category: Tuning

- 2030$_h$: Pole Pair Count
- 2031$_h$: Max Motor Current
- 203B$_h$: I2t Parameters

- 3203$_h$: Feedback Selection
- 3380$_h$: Feedback Sensorless
- 33A0$_h$: Feedback Incremental A/B/I 1
- 6075$_h$: Motor Rated Current
- 608F$_h$: Position Encoder Resolution
- 6090$_h$: Velocity Encoder Resolution
- 60E6$_h$: Additional Position Encoder Resolution - Encoder Increments
- 60EB$_h$: Additional Position Encoder Resolution - Motor Revolutions

7.3.7 Category: Modbus RTU

- 2028$_h$: MODBUS Slave Address
- 202A$_h$: MODBUS RTU Baudrate
- 202D$_h$: MODBUS RTU Parity

7.3.8 Starting the save process

CAUTION!

Uncontrolled motor movements!

Control may be affected while saving. Unforeseen reactions can result.

► The motor must be at a standstill before starting the saving process. The motor must not be started while saving.

NOTICE

- Saving may take a few seconds. Never interrupt the power supply while saving. The state of the saved objects is otherwise undefined.
- Always wait until the controller has signaled that the save process has been successfully completed with the value "1" in the corresponding subindex in object 1010$_h$.

There is a subindex in object 1010$_h$ for each *category*. To save all objects of this *category*, the value "65766173$_h$" must be written in the subindex. [1] The controller signals the end of the save process by overwriting the value with a "1".

The following table shows which subindex of object 1010$_h$ is responsible for which *category*.

Subindex	Category
01$_h$	All categories with the exception of 0B$_h$ (Modbus RTU)
02$_h$	Communication
03$_h$	Application
04$_h$	User
05$_h$	Movement
06$_h$	Tuning
0B$_h$	Modbus RTU

[1] This corresponds to the decimal of `1702257011`$_d$ or the ASCII string `save`.

7.3.9 Discarding the saved data

If all objects or one *category* of saved objects is to be deleted, value "64616F6C$_h$" must be written in object 1011$_h$. [2]

The following subindices correspond to a *category* here:

Subindex	Category
01$_h$	All categories (reset to factory settings) with the exception of 06$_h$ (Tuning) and 0B$_h$ (Modbus RTU)
02$_h$	Communication
03$_h$	Application
04$_h$	User
05$_h$	Movement
06$_h$	Tuning
0B$_h$	Modbus RTU

The saved objects are subsequently discarded; the change does not take effect until after the controller is restarted. You can restart the controller by entering the value "746F6F62$_h$" in 2800$_h$:01$_h$.

NOTICE

- Objects of *category* 06$_h$ (Tuning) are determined by Auto setup and are not reset when resetting to factory settings with subindex 01$_h$ (thereby making it unnecessary to again perform an auto setup). You can reset these objects with subindex 06$_h$.
- Objects of *category* 0B$_h$ (Modbus RTU) are not reset with subindex 01$_h$.

7.3.10 Verifying the configuration

Object 1020$_h$ can be used to verify the configuration. It acts as a modification marker similar to common text editors: as soon as a file is modified in the editor, a marker (usually an asterisk) is added.

The entries of object 1020$_h$ can be written with a date and time and then saved together with all other savable objects with 1010$_h$:01.

The entries of 1020$_h$ are reset to "0" as soon as a savable object (including 1010$_h$:0x$_h$, except for 1010$_h$:01$_h$ and 1020$_h$) is written.

The following sequence makes verification possible:

1. An external tool or master configures the controller.
2. The tool or master sets the value in object 1020$_h$.
3. The tool or master activates the saving of all objects 1010$_h$:01$_h$ = 65766173$_h$. The date and time in object 1020$_h$ are also saved.

After the controller is restarted, the master can check the value in 1020$_h$:01$_h$ and 1020:01$_h$. If one of the values is "0", the object dictionary was changed after the saved values were loaded. If the date or time in 1020 does not correspond to the expected value, objects were probably saved with values other than those that were expected.

7.4 Generic SPI

The controller can communicate with external devices via this SPI interface, e.g. port expanders or displays. The corresponding pins are only available with product variants PD1-...-OF-....

[2] This corresponds to the decimal of `1684107116`$_d$ or the ASCII string `load`.

NOTICE

The used SPI peripheral must support a clock frequency of at least 164 KHz.

The settings of the SPI interface are located in object $3273_h:01_h$ (Generic SPI Hardware Configuration):

- Bit 0 (Clock Phase):
  - Value = "0": Data transfer begins with the first clock edge after *Chip Select* was pulled to low.
  - Value = "1": Data transfer begins with the second clock edge after *Chip Select* was pulled to low.
- Bit 1 (Clock Polarity): With this bit, you can invert the polarity of the clock signal. The value 0 means the level remains on Low if the clock is idling.
- Bits 2 to 4 (baud rate): You set the clock frequency here:
  - 000_b: 21 MHz
  - 001_b: 10.5 MHz
  - 010_b: 5.25 MHz
  - 011_b: 2625 KHz
  - 100_b: 1312.5 KHz
  - 101_b: 656.25 KHz
  - 110_b: 328.125 KHz
  - 111_b: 164.0625 KHz
- Bit 10 (CS Polarity): With this bit, you can invert the polarity of the *Chip Select*. Value 0 means that the level remains on High if the signal is idling.

The data are sent/received via the following objects:

- 3274_h (Generic SPI Mosi Data):
  - Subindex 1 to $1F_h$ (Generic SPI Mosi Data Byte #1 to #31): You write the data that are to be sent here, divided into up to 31 bytes.
  - Subindex 0 (Length of SPI message to be sent): Then enter the number of bytes here (= subindicies) that are to be sent. In the next millisecond cycle, the data are sent and the subindex is reset to the value "0".
- 3275_h (Generic SPI Miso Data): You read the received data here.
  - Subindex 0 (Length of received SPI message): The value tells you how many data bytes (= subindices) were received.
  - Subindex 1 to $1F_h$ (Generic SPI Miso Data Byte #1 to #31): The data that were received are located here.

8 Modbus RTU

Modbus references: www.modbus.org.

- *MODBUS APPLICATION PROTOCOL SPECIFICATION V1.1b3*, Date: 26.04.2014, Version: 1.1b3
- *MODBUS over Serial Line Specification and Implementation Guide V1.02*, Date: 20.12.2006, Version: 1.02

The controller can be controlled by means of Modbus RTU. The I/O data, with, e.g., the preconfigured drive values (see Process data objects (PDO)), can be handled with the standard Modbus function codes. To configure your own I/O data, however, function code 2Bh (CAN Encapsulation) must be supported by the master in order for the parameters to be read and written independent of the process image.

If the master does not support this function code, the I/O image can be configured and stored using *Plug & Drive Studio*. The master can then access the data using the standard Modbus function codes.

8.1 Modbus Modicon notation with PLCs

Many PLCs use the Modicon addressing model. This notation is not used in the Modbus standard.

The following address notation is relevant for Nanotec controllers:

- Input register 30001 - 39999 is mapped to Modbus telegram address 0 (0_h) - 9998 ($270E_h$).
- Holding register 40001 - 49999 is mapped to Modbus telegram address 0 (0_h) - 9998 ($270E_h$).

NOTICE

Where Modbus addresses are mentioned in the manual, it may be necessary to implement the register addresses in the PLC in accordance with *Modicon notation*.

8.2 General

Modbus is generally big-endian based.

The only exceptions are the commands with function codes 43 ($2B_h$), 101 (65_h) and 102 (66_h), which are based on CANopen. For the data values of these commands, the little-endian format applies. The remainder of the Modbus message is, on the other hand, based on big-endian.

Example

Command $2B_h$: With this command, the value `12345678`$_h$ is written in object 0123_h (does not exist):

SA	FC	Data	CRC
05	2B	0D 01 00 01 23 01 00 00 00 00 04 **78 56 34 12**	67 35

SA

Slave address

FC

Function code

Data

Data range, decoding is dependent on the used function code

CRC

Cyclic redundancy check

NOTICE

In case of more than one Modbus slaves in the network, the Modbus master must wait for at least 3 ms after receiving a response before sending the next request.

TIP

To send a broadcast message to all nodes, use slave address "0". The controller does not respond in this case.

8.3 Communication settings

Configuration	Object	Value range	Factory settings
Slave address	2028_h	1 to 247	5
Baud rate	$202A_h$	7200 to 256000	19200
Parity	$202D_h$	■ None: 0x00 ■ Even: 0x04 ■ Odd: 0x06	0x04 (Even)

The number of data bits is always "8" here. The number of stop bits is dependent on the parity setting:

- No parity: 2 stop bits
- "Even" or "Odd" parity: 1 stop bit

The following baud rates are supported:

- 7200
- 9600
- 14400
- 19200
- 38400
- 56000
- 57600
- 115200
- 128000
- 256000

You must save the changes by writing value "65766173_h" in object 1010_h:$0B_h$. The changes are not taken over until after the controller has been restarted.

8.4 Function codes

The following "function codes" are supported:

	Name	Function code	Subfunction code
Data access (16-bit)	Read Holding Registers	03 (03_h)	
	Read Input Register	04 (04_h)	
	Write Single Register	06 (06_h)	
	Write Multiple Registers	16 (10_h)	

Name		Function code	Subfunction code
	Read/Write Multiple Registers	23 (17_h)	
Diagnosis	Clear Counters and Diagnostic Register	08 (08_h)	10 ($0A_h$)
	Return Bus Message Count	08 (08_h)	11 ($0B_h$)
	Return Bus Communication Error Count	08 (08_h)	12 ($0C_h$)
	Return Bus Exception Error Count	08 (08_h)	13 ($0D_h$)
	Return Server Message Count	08 (08_h)	14 ($0E_h$)
	Return Server No Response Count	08 (08_h)	15 ($0F_h$)
	Return Server NAK Count	08 (08_h)	16 (10_h)
	Return Server Busy Count	08 (08_h)	17 (11_h)
	Return Bus Character Overrun Count	08 (08_h)	18 (12_h)
Miscellaneous	Encapsulated Interface Transport	43 ($2B_h$)	13 ($0D_h$)
	Read complete object dictionary start	101 (65_h)	85 (55_h)
	Read complete object dictionary next	101 (65_h)	170 (AA_h)
	Read complete array or record start	102 (66_h)	85 (55_h)
	Read complete array or record next	102 (66_h)	170 (AA_h)

8.5 Function code descriptions

8.5.1 FC 3 (03_h) Read Input Registers / FC 4 (04_h) Read Holding Registers

With this function code, one 16-bit value or multiple 16-bit values can be read. This function can be applied to NanoJ objects (see NanoJ objects) or process data objects (min. 4-byte alignment, see Process data objects (PDO)).

Request		
Name	**Length**	**Value**
Slave address	1 byte	
Function code	1 byte	03_h / 04_h
Start address	2 bytes	0000_h to $FFFF_h$
Number of registers	2 bytes	1 to ($7D_h$)
CRC	2 bytes	

Response ("M" corresponds to the number of registers to be read)		
Name	**Length**	**Value**
Slave address	1 byte	
Function code	1 byte	03_h / 04_h
Number of bytes	1 byte	2 * M
Register value	2 bytes	
CRC	2 bytes	

Error		
Name	**Length**	**Value**
Slave address	1 byte	

Error		
Name	**Length**	**Value**
Error code	1 byte	83_h / 84_h
Exception code (see Exception codes)	1 byte	01, 02, 03 or 04
CRC	2 bytes	

Example

Below is an example of a read request and response of register 5000 (1388_h) and of the following register (2 registers):

Request

SA	FC	Data	CRC
05	03	13 88 00 02	41 21

Response

SA	FC	Data	CRC
05	03	04 02 40 00 00	41 21

8.5.2 FC 6 (06_h) Write Single Register

This function code can be used to write a single 16-bit value. The function can be used on process data objects (see Process data objects (PDO)).

Request		
Name	**Length**	**Value**
Slave address	1 byte	
Function code	1 byte	06_h
Register address	2 bytes	0000_h to $FFFF_h$
Register value	2 bytes	0000_h to $FFFF_h$
CRC	2 bytes	

Response		
Name	**Length**	**Value**
Slave address	1 byte	
Function code	1 byte	06_h
Register address	2 bytes	0000_h to $FFFF_h$
Register value	2 bytes	0000_h to $FFFF_h$
CRC	2 bytes	

Error		
Name	**Length**	**Value**
Slave address	1 byte	

Error		
Name	**Length**	**Value**
Error code	1 byte	86_h
Exception code (see Exception codes)	1 byte	01, 02, 03 or 04
CRC	2 bytes	

Example

Below is an example of a write request and response in register 6000 (1770_h) with the value "0001_h":

Request

SA	FC	Data	CRC
05	06	17 70 00 01	4D E1

Response

SA	FC	Data	CRC
05	06	17 70 00 01	4D E1

8.5.3 FC 16 (10_h) Write Multiple Registers

With this function code, one 16-bit value or multiple 16-bit values can be written. The function can be applied to NanoJ objects (see Process data objects (PDO)) or process data objects (see NanoJ objects).

Request ("N" is the number of registers to be written)		
Name	**Length**	**Value**
Slave address	1 byte	
Function code	1 byte	10_h
Start address	2 bytes	0000_h to $FFFF_h$
Number of registers	2 bytes	0001_h to $007B_h$
Number of bytes	1 byte	2 * N
Register value	N * 2 bytes	
CRC	2 bytes	

Response		
Name	**Length**	**Value**
Slave address	1 byte	
Function code	1 byte	10_h
Start address	2 bytes	0000_h to $FFFF_h$
Number of registers	2 bytes	0001_h to $007B_h$
CRC	2 bytes	

Error		
Name	**Length**	**Value**
Slave address	1 byte	
Error code	1 byte	90_h
Exception code (see Exception codes)	1 byte	01, 02, 03 or 04
CRC	2 bytes	

Example

Below is an example for writing values "0102_h" and "0304_h" starting with register address 6000 (1770_h), number of registers is 2, length of the data is 4:

Request

SA	FC	Data	CRC
05	10	17 70 00 02 04 01 02 03 04	AB 44

Response

SA	FC	Data	CRC
05	10	17 70 00 02	44 23

8.5.4 FC 17 (11_h) Report Server ID

This function code can be used to read the description of the type, the current status and other information about the device.

Request		
Name	**Length**	**Value**
Slave address	1 byte	
Function code	1 byte	11_h
CRC	2 bytes	

Response		
Name	**Length**	**Value**
Slave address	1 byte	
Function code	1 byte	03_h
Number of bytes	1 byte	01_h
Run Indicator Status	1 byte	00_h = OFF, FF_h = ON
Additional data		
CRC	2 bytes	

Error		
Name	**Length**	**Value**
Slave address	1 byte	

Error		
Name	**Length**	**Value**
Error code	1 byte	91_h
Exception code (see Exception codes)	1 byte	01 or 04
CRC	2 bytes	

Example

Below is an example of a request/response for ID and status:

Request

SA	FC	CRC
05	11	C2 EC

Response

SA	FC	Data	CRC
05	11	02 05 FF	0F EC

8.5.5 FC 23 (17_h) Read/Write Multiple registers

With this function code, one 16-bit value or multiple 16-bit values can be simultaneously read and written. The function can be applied to NanoJ objects (see Process data objects (PDO)) or process data objects (see NanoJ objects).

Request ("N" is the number of registers to be read):		
Name	**Length**	**Value**
Slave address	1 byte	
Function code	1 byte	17_h
Read: Start address	2 bytes	0000_h to $FFFF_h$
Read: Number of registers	2 bytes	0001_h to 0079_h
Write: Start address	2 bytes	0000_h to $FFFF_h$
Write: Number of registers	2 bytes	0001_h to 0079_h
Write: Number of bytes	1 byte	2 * N
Write: Register value	N * 2 bytes	
CRC	2 bytes	

Response ("M" corresponds to the number of bytes to be written):		
Name	**Length**	**Value**
Slave address	1 byte	
Function code	1 byte	17_h
Number of bytes	1 byte	2 * M
Registers read	M * 2 bytes	
CRC	2 bytes	

Error		
Name	**Length**	**Value**
Slave address	1 byte	
Error code	1 byte	97_h
Exception code (see Exception codes)	1 byte	01, 02, 03 or 04
CRC	2 bytes	

Example

Below is an example for reading two registers beginning with register 5000 (1388_h) and for writing two registers beginning with register 6000 (1770_h) with 4 bytes and data "0102_h" and "0304_h":

Request

SA	FC	Data	CRC
05	17	13 88 00 02 17 70 00 02 04 01 02 03 04	56 6A

Response

SA	FC	Data	CRC
05	17	04 02 40 00 00	0F EC

8.5.6 FC 8 (08_h) Diagnostics

Modbus function code FC08 offers numerous tests for checking the communication system between client and server or for checking various internal error states within the server.

This function uses a two-byte subfunction code in the request for defining the type of test. In a normal response, the server repeats both, the function and the subfunction code. Some diagnoses contain data of the device in the data field of the normal response.

Request:

Name	Length	Value
Function code	1 byte	08_h
Subfunction code	2 bytes	
Data	N x 2 bytes	

Response:

Name	Length	Value
Function code	1 byte	08_h
Subfunction code	2 bytes	
Data	N x 2 bytes	

Error:

Name	Length	Value
Function code	1 byte	88_h
Exception code (see Exception codes)	1 byte	01 or 03 or 04

8.5.6.1 FC 8.10 (08_h.$0A_h$) Clear Counters and Diagnostic Register

The objective of this request is to reset all counters and diagnosis registers. Counters are also reset when the controller is switched on.

Subfunction	Data range	
	Request	Response
00_h $0A_h$	00_h - 00_h	Echo of the request data

Example

Request

SA	FC	Data	CRC
05	08	00 0A 00 00	56 6A

Response

SA	FC	Data	CRC
05	08	00 0A 00 00	C1 8D

8.5.6.2 FC 8.11 (08_h.$0B_h$) Return Bus Message Count

The response data range returns the number of messages detected by the communications system since the last restart, "Clear Counters and Diagnostic Register" request, or switching on of the controller.

Subfunction	Data range	
	Request	Response
00_h $0B_h$	00_h - 00_h	Total Message Count

8.5.6.3 FC 8.12 (08_h.$0C_h$) Return Bus Communication Error Count

The response data range returns the number of CRC errors since the last restart, "Clear Counters and Diagnostic Register" request, or switching on of the controller.

Subfunction	Data range	
	Request	Response
00_h $0C_h$	00_h - 00_h	CRC Error Count

Example

Request

SA	FC	Data	CRC
05	08	00 0C 00 00	21 8C

Response

SA	FC	Data	CRC
05	08	00 0C 00 00	21 8C

8.5.6.4 FC 8.13 (08_h.$0D_h$) Return Bus Exception Error Count

The response data range returns the number of Modbus exceptions since the last restart, "Clear Counters and Diagnostic Register" request, or switching on of the controller.

Subfunction	Data range	
	Request	Response
00_h $0D_h$	00_h - 00_h	Exception Error Count

Example

Request

SA	FC	Data	CRC
05	08	00 0D 00 00	70 4C

Response

SA	FC	Data	CRC
05	08	00 0D 00 00	70 4C

8.5.6.5 FC 8.14 (08_h.$0E_h$) Return Server Message Count

The response data range returns the number of messages addressed to the device and the number of broadcast messages that were processed by the controller. The number of messages since the last restart, "Clear Counters and Diagnostic Register" request, or switching on of the controller are counted.

Subfunction	Data range	
	Request	Response
00_h $0E_h$	00_h - 00_h	Server Message Count

Example

Request

SA	FC	Data	CRC
05	08	00 0E 00 00	80 4C

Response

SA	FC	Data	CRC
05	08	00 0E 00 00	80 4C

8.5.6.6 FC 8.15 (08_h.$0F_h$) Return Server No Response Count

The response data range returns the number of messages addressed to the controller for which no response was returned (neither normal response nor exception response). The number of messages since the last restart, "Clear Counters and Diagnostic Register" request, or switching on of the controller are counted.

Subfunction	Data range	
	Request	Response
00_h $0F_h$	00_h - 00_h	No Response Count

Example

Request

SA	FC	Data	CRC
05	08	00 0F 00 00	D1 8C

Response

SA	FC	Data	CRC
05	08	00 0F 00 00	D1 8C

8.5.6.7 FC 8.16 (08_h.10_h) Return Server NAK Count

The response data range returns the number of messages for which a "Negative Acknowledge (NAK)" exception response was returned. The number of messages since the last restart, "Clear Counters and Diagnostic Register" request, or switching on of the controller are counted.

Subfunction	Data range	
	Request	Response
00_h - 10_h	00_h - 00_h	Server NAK Count

Example

Request

SA	FC	Data	CRC
05	08	00 10 00 00	E0 4A

Response

SA	FC	Data	CRC
05	08	00 10 00 00	E0 4A

8.5.6.8 FC 8.17 (08_h.11_h) Return Server Busy Count

The response data range returns the number of messages for which a "Server Device Busy" exception response was returned. The number of messages since the last restart, "Clear Counters and Diagnostic Register" request, or switching on of the controller are counted.

Subfunction	Data range	
	Request	Response
00_h - 11_h	00_h - 00_h	Server NAK Count

Example

Request

SA	FC	Data	CRC
05	08	00 11 00 00	B1 8A

Response

SA	FC	Data	CRC
05	08	00 11 00 00	B1 8A

8.5.6.9 FC 8.18 (08_h.12_h) Return Bus Character Overrun Count

The response data range returns the number of messages addressed to the controller that could not be processed due to a character overrun. The number of messages since the last restart, "Clear Counters and Diagnostic Register" request, or switching on of the controller are counted. A character overrun occurs when characters arrive at the controller faster than they can be stored or by the loss of a character due to a hardware malfunction.

Subfunction	Data range	
	Request	Response
00_h - 12_h	00_h - 00_h	Server Character Overrun Count

Example

Request

SA	FC	Data	CRC
05	08	00 12 00 00	41 8A

Response

SA	FC	Data	CRC
05	08	00 12 00 00	41 8A

8.5.7 FC 43 ($2B_h$) Encapsulated Interface Transport

This function facilitates simple access of the CANopen object dictionary. Further details can be found in the following documentation:

1. *MODBUS APPLICATION PROTOCOL SPECIFICATION V1.1b3*, Date: 26.04.2014, Version: 1.1b3
2. *CiA 309 Draft Standard Proposal - Access from other networks - Part 2: Modbus/TCP mapping V1.3*, Date: 30.07.2015, Version: 1.3

NOTICE

For the messages of the Encapsulated Interface Transport, another byte sequence applies in part, see chapter General.

Definition of the request and response:

Name	Length	Example/number range
Slave address	1 byte	
Function code	1 byte	$2B_h$ (43_d)
MEI type	1 byte	$0D_h$ (13_d)
Protocol options Range	2 to 5 bytes	
Address and data range	N bytes	
CRC	2 bytes	

Protocol options Range

Name	Length	Example/number range
Protocol control	1 to 2 bytes	See description
Reserved	1 byte	Always 0
(Optional) Counter byte	1 byte	
(Optional) Network ID	1 byte	
(Optional) Encoded data	1 byte	

Protocol control:

The "Protocol control" field contains the flags that are needed for controlling the message protocols. The bytes of the "Protocol control" field are defined as follows if the "extended" flag was set (the second byte is otherwise omitted):

0	7	8	15
Protocol control byte 1		Protocol control byte 2	
MSB	LSB	MSB	LSB

The most significant bit (MSB) is bit 0 for "protocol control" byte 1 and bit 8 for "protocol control" byte 2. The least significant bit (LSB) is bit 7 for "protocol control" byte 1 and bit 15 for "protocol control" byte 2.

Bit	Name	Description
0	"Extended" flag	This bit is used if the object dictionary data set is larger than would fit in a Modbus command. The data set then spans over multiple Modbus messages; each message contains part of the data set. "0" = No multiple message transaction or the end of the multiple message transaction. "1" = Part of a multiple message transaction.
1	Extended protocol control	Length of the protocol control, the value "0" indicates a length of 1 byte, the value "1" indicates a length of 2 bytes.
2	Counter byte option	This bit is set to "1" to indicate that the "counter byte" field is used in this message. If this bit is set to "0", the "counter byte" field does not exist in this message.
3 and 4	Reserved	0
5	Network ID option	Not supported, must be "0".
6	Encoded data option	Not supported, must be "0".
7	Access flag	This bit indicates the access method of the requested command. "0" = read, "1" = write.
8 to 15	Reserved	0

Address and data range

The address and data range is defined in the following table:

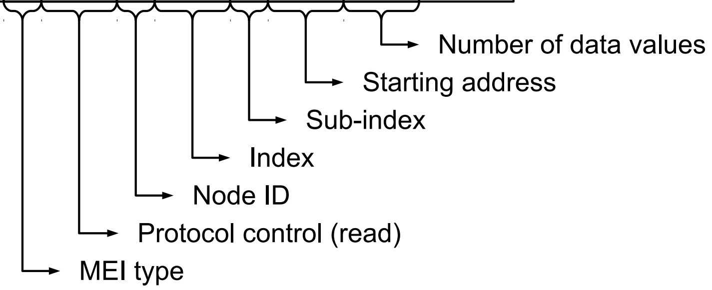
Name	Byte size and byte order	Example / range
Node-ID	1 byte	01_h to $7F_h$
Index	1 byte, high 1 byte, low	0000_h to $FFFF_h$
Subindex	1 byte	00_h to FF_h
Start address	1 byte, high 1 byte, low	0000_h to $FFFF_h$
Number of data values	1 byte, high 1 byte, low	0000_h to $00FD_h$
Write/read data	n bytes	The data are encoded as described in chapter General.

Example:

To read object 6042$_h$:00$_h$ (16-bit value), the following message must be sent by the master (all values are in hexadecimal notation, the slave ID of the controller is "5").

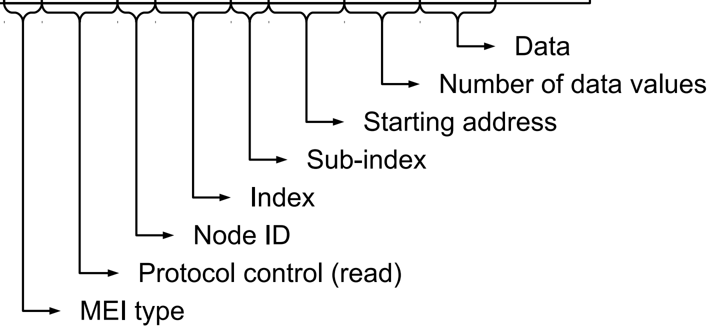
Request

SA	FC	Data							CRC
05	2B	0D	00 00	01	60 42	00	00 00	00 02	7F 0F



Response

SA	FC	Data								CRC
05	2B	0D	00 00	01	60 42	00	00 00	00 02	00 00	60 34



Shown as an additional example below, a sequence of Modbus messages is sent from the master to the slave to rotate the motor in "Velocity" mode:

Set 6060 = "02$_h$" (Velocity mode)

Request

SA	FC	Data	CRC
05	2B	0D 01 00 01 60 60 00 00 00 00 01 02	C9 2F

Response

SA	FC	Data	CRC
05	2B	0D 01 00 01 60 60 00 00 00 00 00	A9 89

Set 2031 = 03E8$_h$" (1000 mA)

Request

SA	FC	Data	CRC
05	2B	0D 01 00 01 20 31 00 00 00 00 04 E8 03 00 00	C3 53

Response

SA	FC	Data	CRC
05	2B	0D 01 00 01 20 31 00 00 00 00 00	E5 CC

Set 6040 = "00_h"

Request

SA	FC	Data	CRC
05	2B	0D 01 00 01 60 40 00 00 00 00 02 00 00	1C 2E

Response

SA	FC	Data	CRC
05	2B	0D 01 00 01 60 40 00 00 00 00 00	AE E9

Set 6040 = "80_h"

Request

SA	FC	Data	CRC
05	2B	0D 01 00 01 60 40 00 00 00 00 02 80 00	7D EE

Response

SA	FC	Data	CRC
05	2B	0D 01 00 01 60 40 00 00 00 00 00	AE E9

Set 6040 = "06_h"

Request

SA	FC	Data	CRC
05	2B	0D 01 00 01 60 40 00 00 00 00 02 06 00	1F 8E

Response

SA	FC	Data	CRC
05	2B	0D 01 00 01 60 40 00 00 00 00 00	AE E9

Set 6040 = "07_h"

Request

SA	FC	Data	CRC
05	2B	0D 01 00 01 60 40 00 00 00 00 02 07 00	1E 1E

Response

SA	FC	Data	CRC
05	2B	0D 01 00 01 60 40 00 00 00 00 00	AE E9

Set 6040 = "0F$_h$"

Request

SA	FC	Data	CRC
05	2B	0D 01 00 01 60 40 00 00 00 00 02 0F 00	19 DE

Response

SA	FC	Data	CRC
05	2B	0D 01 00 01 60 40 00 00 00 00 00	AE E9

Below are two examples for reading an object:

Read 6041$_h$:00$_h$

Request

SA	FC	Data	CRC
05	2B	0D 00 00 01 60 41 00 00 00 00 02	7F 3C

Response

SA	FC	Data	CRC
05	2B	0D 00 00 01 60 41 00 00 00 00 02 37 06	B6 13

Read 6061$_h$:00$_h$

Request

SA	FC	Data	CRC
05	2B	0D 00 00 01 60 61 00 00 00 00 01	38 5D

Response

SI	FC	Data	CRC
05	2B	0D 00 00 01 60 61 00 00 00 00 01 00	5C D2

8.5.7.1 Error reaction

In the event of an error, the following error message is sent:

Name	Length	Example value
Function code	1 byte	$2B_h$ +80_h (171_d = 43_d + 128_d) (indicates error)

Name	Length	Example value
Modbus exception code	1 byte	FF_h ("extended exception")
Extended exception length	2 bytes	6
MEI type	1 byte	$0D_h$
Exception code	1 byte	CE_h
Error code	4 bytes	CANopen error code, see following table

CANopen error code	Description
$FFFF0000_h$	*Abort no error*
$FFFF1003_h$	Service is not supported
$FFFF1004_h$	Gap in counter byte of the *Protocol control* field
$FFFF0003_h$	Unknown or invalid command
$FFFF0008_h$	Access to the object is not supported
$FFFF000E_h$	General error in the parameter
$FFFF0011_h$	Length of parameter incorrect
$FFFF0012_h$	Parameter too long
$FFFF0013_h$	Parameter too short
$FFFF0015_h$	Parameter data outside of the permissible value range (for write commands)
$FFFF0016_h$	Parameter data exceed the permissible value range (for write commands)
$FFFF0017_h$	Parameter data below the permissible value range (for write commands)
$FFFF0018_h$	Maximum entered values less than minimum values
$FFFF0019_h$	General error
$FFFF001E_h$	Requested object is too large for single message
$FFFF1004_h$	Invalid sequence of messages (e. g., if the value of the *counter byte* is not correct according to the previous request or response)

In the event that the unsupported control option bit is set, the following error message is sent:

Name	Length	Example value
Function code	1 byte	$2B_h$ +80_h (171_d = 43_d + 128_d) (indicates error)
Modbus exception code	1 byte	FF_h ("extended exception")
Extended exception length	2 bytes	2 + length of "supported protocol control"
MEI type	1 byte	$0D_h$
Exception code	1 byte	AE_h
Supported protocol control	1 or 2 bytes	See following table

Bit	Name	Description
0	"Extended" flag	This bit is used if the object dictionary data set is larger than would fit in a Modbus command. The data set then spans over multiple Modbus messages; each message contains part of the data set. "0" = No multiple message transaction or the end of the multiple

Bit	Name	Description
		message transaction. "1" = Part of a multiple message transaction.
1	Extended protocol control	Length of the protocol control, the value "0" indicates a length of 1 byte, the value "1" indicates a length of 2 bytes.
2	Counter byte option	This bit is set to "1" to indicate that the "counter byte" field is used in this message. If this bit is set to "0", the "counter byte" field does not exist in this message.
3 and 4	Reserved	0
5	Network ID option	Not supported, must be "0".
6	Encoded data option	Not supported, must be "0".
7	Access flag	This bit indicates the access method of the requested command. "0" = read, "1" = write.
8 to 15	Reserved	0

8.5.8 FC 101 (65_h) Read complete object dictionary

This function code is used to read out the complete object dictionary.

To start or restart the reading out of the object dictionary, subfunction code `55`$_h$ must be sent. This code resets reading out of the object dictionary on object 0000_h. All subsequent object dictionary frames must then contain subfunction code `AA`$_h$. At the end, once all objects have been read out, an "Error Response" is generated with the abort code "No data available".

The format of each "read object" is as follows:

Request:

Name	Length	Value / note
Slave address	1 byte	
Function code	1 byte	65_h
Subfunction code	1 byte	55_h or AA_h
Length of the data	1 byte	00_h
CRC	2 bytes	

Response:

Name	Length	Value / note
Slave address	1 byte	65_h
Function code	1 byte	
Subfunction code	1 byte	
Length of the data	1 byte	
n times "object dictionary frame"	1 - 252 bytes	
CRC	2 bytes	

An object dictionary frame consists of the following bytes:

Name		Value / note
Index Low Byte	1 byte	

Name	Value / note	
Index High Byte	1 byte	
Subindex	1 byte	
Number of bytes	1 byte	Number m of the valid data in the data field
Data byte	m-1 byte	

Example

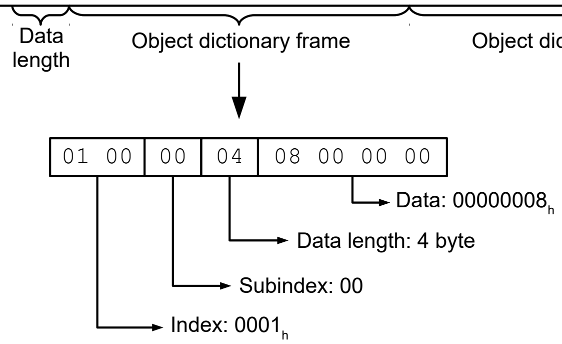
All of the following numerical values are in hexadecimal format. The address of the slave is "5".

Start reading of the object dictionary with request:

SA	FC	Data	CRC
05	65	55 00	2F A7

The response is:

SA	FC	Data	CRC
05	65	55 CE 01 00 00 04 08 00 00 00 02 00 00 04 08 00 00 00 ...	EA 3A



Read out the next part of the object dictionary with the request:

SA	FC	Data	CRC
05	65	AA 00	6E 57

The response is:

SA	FC	Data	CRC
05	65	AA CD 21 00 0A 02 07 00 21 00 0B 02 07 00 21 00 0C 02 ...	NN NN

Repeat reading of the object dictionary with the previous request until the response is an error:

SA	FC	Data	CRC
05	E5	0D	EA 94

8.5.8.1 Error reaction

In the event of an error, the following error message is sent:

Name	Length	Example value
Function code	1 byte	$2B_h$ +80_h (171_d = 43_d + 128_d) (indicates error)
Modbus exception code	1 byte	FF_h ("extended exception")
Extended exception length	2 bytes	6
MEI type	1 byte	$0D_h$
Exception code	1 byte	CE_h
Error code	4 bytes	CANopen error code, see following table

CANopen error code	Description
$FFFF0000_h$	*Abort no error*
$FFFF1003_h$	Service is not supported
$FFFF1004_h$	Gap in counter byte of the *Protocol control* field
$FFFF0003_h$	Unknown or invalid command
$FFFF0008_h$	Access to the object is not supported
$FFFF000E_h$	General error in the parameter
$FFFF0011_h$	Length of parameter incorrect
$FFFF0012_h$	Parameter too long
$FFFF0013_h$	Parameter too short
$FFFF0015_h$	Parameter data outside of the permissible value range (for write commands)
$FFFF0016_h$	Parameter data exceed the permissible value range (for write commands)
$FFFF0017_h$	Parameter data below the permissible value range (for write commands)
$FFFF0018_h$	Maximum entered values less than minimum values
$FFFF0019_h$	General error
$FFFF001E_h$	Requested object is too large for single message
$FFFF1004_h$	Invalid sequence of messages (e. g., if the value of the *counter byte* is not correct according to the previous request or response)

In the event that the unsupported control option bit is set, the following error message is sent:

Name	Length	Example value
Function code	1 byte	$2B_h$ +80_h (171_d = 43_d + 128_d) (indicates error)
Modbus exception code	1 byte	FF_h ("extended exception")
Extended exception length	2 bytes	2 + length of "supported protocol control"
MEI type	1 byte	$0D_h$
Exception code	1 byte	AE_h
Supported protocol control	1 or 2 bytes	See following table

Bit	Name	Description
0	"Extended" flag	This bit is used if the object dictionary data set is larger than would fit in a Modbus command. The

Bit	Name	Description
		data set then spans over multiple Modbus messages; each message contains part of the data set. "0" = No multiple message transaction or the end of the multiple message transaction. "1" = Part of a multiple message transaction.
1	Extended protocol control	Length of the protocol control, the value "0" indicates a length of 1 byte, the value "1" indicates a length of 2 bytes.
2	Counter byte option	This bit is set to "1" to indicate that the "counter byte" field is used in this message. If this bit is set to "0", the "counter byte" field does not exist in this message.
3 and 4	Reserved	0
5	Network ID option	Not supported, must be "0".
6	Encoded data option	Not supported, must be "0".
7	Access flag	This bit indicates the access method of the requested command. "0" = read, "1" = write.
8 to 15	Reserved	0

8.5.9 FC 102 (66_h) Read complete array or record

This function code is used to read out the complete array or record from the object dictionary.

To start or restart the reading out of the array, subfunction code `55`$_h$ must be sent. This code resets reading out on the object with subindex 00_h. All subsequent requests must then contain subfunction code `AA`$_h$. At the end, once all objects have been read out, an "Error Response" is generated.

The format of each "read object" is as follows:

Request:

Name	Length	Value / note
Slave address	1 byte	
Function code	1 byte	66_h
Subfunction code	1 byte	55_h or AA_h
Length of the data	1 byte	00_h
Index of the array to be read	2 bytes	
CRC	2 bytes	

Response:

Name	Length	Value / note
Slave address	1 byte	65_h
Function code	1 byte	
Subfunction code	1 byte	
Length of the data	1 byte	
n times object dictionary frame	1 - 252 bytes	
CRC	2 bytes	

An object dictionary frame consists of the following bytes:

Name	Value / note	
Index Low Byte	1 byte	
Index High Byte	1 byte	
Subindex	1 byte	
Number of bytes	1 byte	Number m of the valid data in the data field
Data byte	m-1 byte	

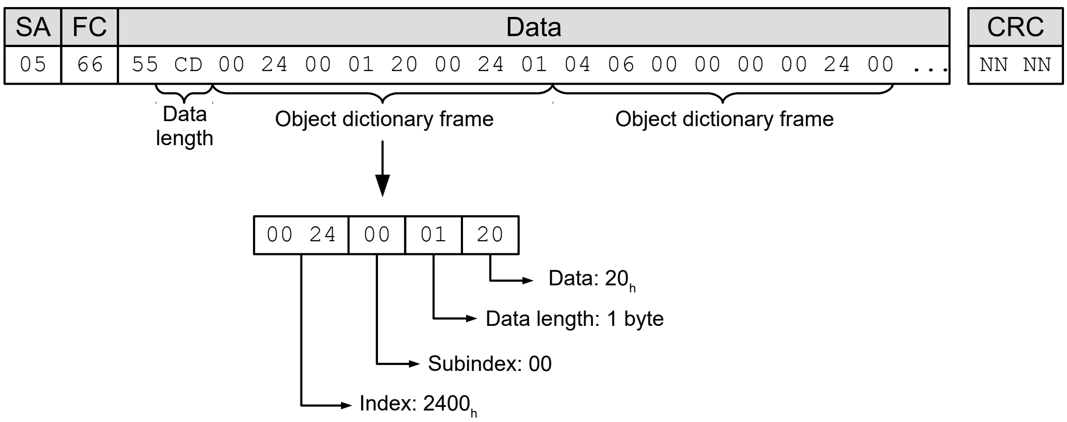
Example

All of the following numerical values are in hexadecimal format; the index of the object that is to be read is 2400_h. The address of the slave is "5"$_h$.

Start reading of the array with request:

SA	FC	Data	CRC
05	66	55 00 24 00	02 8A

The response is:



8.5.9.1 Error reaction

In the event of an error, the following error message is sent:

Name	Length	Example value
Function code	1 byte	$2B_h$ +80_h (171_d = 43_d + 128_d) (indicates error)
Modbus exception code	1 byte	FF_h ("extended exception")
Extended exception length	2 bytes	6
MEI type	1 byte	$0D_h$
Exception code	1 byte	CE_h
Error code	4 bytes	CANopen error code, see following table

CANopen error code	Description
$FFFF0000_h$	*Abort no error*
$FFFF1003_h$	Service is not supported
$FFFF1004_h$	Gap in counter byte of the *Protocol control* field

CANopen error code	Description
$FFFF0003_h$	Unknown or invalid command
$FFFF0008_h$	Access to the object is not supported
$FFFF000E_h$	General error in the parameter
$FFFF0011_h$	Length of parameter incorrect
$FFFF0012_h$	Parameter too long
$FFFF0013_h$	Parameter too short
$FFFF0015_h$	Parameter data outside of the permissible value range (for write commands)
$FFFF0016_h$	Parameter data exceed the permissible value range (for write commands)
$FFFF0017_h$	Parameter data below the permissible value range (for write commands)
$FFFF0018_h$	Maximum entered values less than minimum values
$FFFF0019_h$	General error
$FFFF001E_h$	Requested object is too large for single message
$FFFF1004_h$	Invalid sequence of messages (e. g., if the value of the *counter byte* is not correct according to the previous request or response)

In the event that the unsupported control option bit is set, the following error message is sent:

Name	Length	Example value
Function code	1 byte	$2B_h$ +80_h (171_d = 43_d + 128_d) (indicates error)
Modbus exception code	1 byte	FF_h ("extended exception")
Extended exception length	2 bytes	2 + length of "supported protocol control"
MEI type	1 byte	$0D_h$
Exception code	1 byte	AE_h
Supported protocol control	1 or 2 bytes	See following table

Bit	Name	Description
0	"Extended" flag	This bit is used if the object dictionary data set is larger than would fit in a Modbus command. The data set then spans over multiple Modbus messages; each message contains part of the data set. "0" = No multiple message transaction or the end of the multiple message transaction. "1" = Part of a multiple message transaction.
1	Extended protocol control	Length of the protocol control, the value "0" indicates a length of 1 byte, the value "1" indicates a length of 2 bytes.
2	Counter byte option	This bit is set to "1" to indicate that the "counter byte" field is used in this message. If this bit is set to "0", the "counter byte" field does not exist in this message.
3 and 4	Reserved	0
5	Network ID option	Not supported, must be "0".
6	Encoded data option	Not supported, must be "0".

Bit	Name	Description
7	Access flag	This bit indicates the access method of the requested command. "0" = read, "1" = write.
8 to 15	Reserved	0

8.5.10 Exception codes

In case of an error, the following exception codes may be contained in the response depending on the function code:

Code	Name	Description
01	Illegal Function	Function code not recognized/allowed
02	Illegal Data Address	Register address not valid or does not exist
03	Illegal Data Value	Value not valid
04	Device Failure	Unrecoverable error

For further details, refer to Modbus specification *MODBUS APPLICATION PROTOCOL SPECIFICATION V1.1b3*.

8.6 Process data objects (PDO)

As with CANopen, a process image can be configured for input and output values with Modbus. This image only contains the data values of one or more objects without additional information, such as length, index or subindex. A single message can thereby be used to read or write multiple objects at the same time.

8.6.1 Configuration

The configuration of the image is referred to as "mapping" and is written in the following objects:

- 3502_h for the Modbus Rx (master → slave) PDO mapping
- 3602_h for Modbus Tx (slave → master) PDO mapping

Both objects contain an array of 16 entries each. Subindex 00 specifies the number of valid entries here.

Objects 3502_h and 3602_h can be written with messages with Modbus function code $2B_h$.

8.6.2 Transfer

The data are written sequentially in the message without gaps and alignment.

If alignment is required (e.g., 16-bit alignment), additional "dummy objects" can be incorporated in the message. Dummy objects are only ever transferred with the data value "0". These objects are listed in the following table.

Index	Data type
0002_h	Signed integer (8 bit)
0003_h	Signed integer (16 bit)
0004_h	Signed integer (32 bit)
0005_h	Unsigned integer (8 bit)
0006_h	Unsigned integer (16 bit)
0007_h	Unsigned integer (32 bit)

Mapping is as follows:

- The PDO RX image begins at Modbus register address 6000_d (1770_h).
- The PDO TX image begins at Modbus register address 5000_d (1388_h).

Read/write access can be performed simultaneously with function code 17_h or with the 03_h, 04_h, 06_h, 10_h commands on the respective RX/TX images.

NOTICE

To be able to change the mapping, you must first deactivate it by setting the corresponding subindex 0_h to "0".

After writing the objects to the respective subindices, enter the number of mapped objects in subindex 0_h.

Example

The following objects are to be set in the mapping:

- 3602_h:00_h = "0_h" (mapping is deactivated)
- 3602_h:01_h = "60410010_h" (object 6041_h:00_h, length 16 bits is mapped)
- 3602_h:02_h = "00050008_h" (dummy object 0005_h:00_h, length 8 bits is mapped)
- 3602_h:03_h = "60610008_h" (object 6061_h:00_h, length 8 bits is mapped)
- 3602_h:04_h = "60640020_h" (object 6064_h:00_h, length 32 bits is mapped)
- 3602_h:05_h = "60440010_h" (object 6044_h:00_h, length 16 bits is mapped)
- 3602_h:06_h = "$60FD0020_h$" (object $60FD_h$:00_h, length 32 bits is mapped)
- 3602_h:00_h = "6_h" (6 values are mapped)

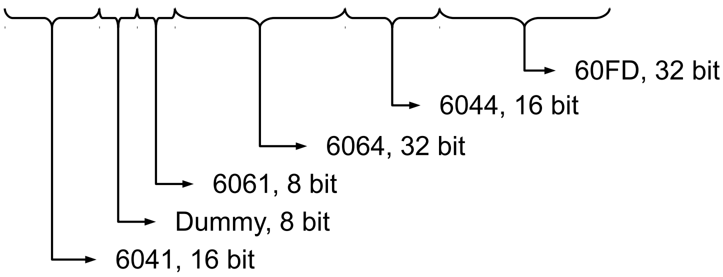
After the mapping for object 6061_h:00_h, a dummy object is inserted so that the next object 6064_h:00_h can be aligned to 32 bit.

Rx message: The master sends the slave the following message:

SA	FC	Data	CRC
05	04	13 88 00 07	34 E2

Tx message: The slave sends following response to the master:

SA	FC	Data							CRC
05	04	0E	06 40	00	02	00 00 00 00	00 00	00 00 00 00	1C 98



8.7 NanoJ objects

NanoJ objects 2400_h NanoJ Input and 2500_h (NanoJ Output) are, like the process image, mapped to the Modbus register:

- 2500_h with 32 x 32 bit values is mapped to the Modbus register address beginning with 2000_d ($7D0_h$) and can only be read in this way.
- 2400_h with 32 x 32 bit values is mapped to the Modbus register address beginning with 3000_d ($BB8_h$) and can only be written in this way.

To access, commands with function codes 03_h, 04_h, 10_h and 17_h can be used. For purposes of data consistency, the restriction that the address must be 32-bit aligned and that at least 32 bits must always be written during a write operation applies.

Example

Request: The master sends the slave the following message:

SA	FC	Data	CRC
05	17	07 D0 00 08 0B B8 00 08 10 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F	41 21

Reply: The slave sends the master the following response:

SA	FC	Data	CRC
05	17	10 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	50 9D

9 Programming with NanoJ

NanoJ is a programming language similar to *C* or *C++*. NanoJ is integrated in the *Plug & Drive Studio* software. You can find further information in the document *Plug & Drive Studio: Quick Start Guide* at us.nanotec.com.

9.1 NanoJ program

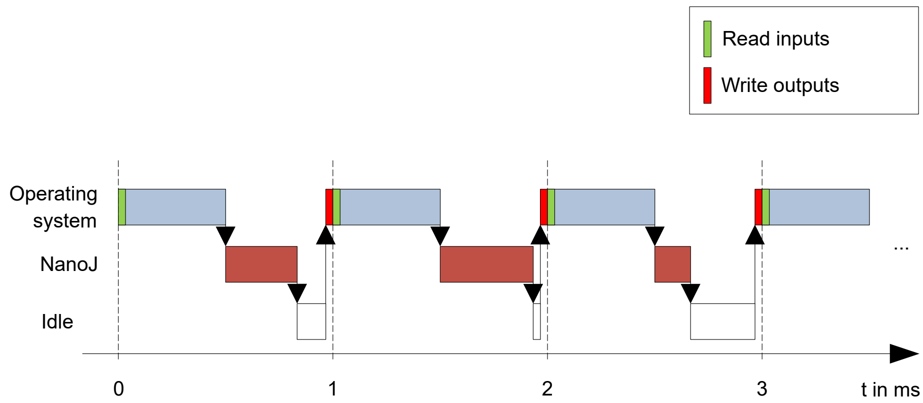
A *NanoJ program* makes a protected runtime environment available within the firmware. Here, the user can create his own processes. These can then trigger functions in the controller by, for example, reading or writing entries in the object dictionary.

Through the use of protective mechanisms, a *NanoJ program* is prevented from crashing the firmware. In the worst case, the execution is interrupted with an error code stored in the object dictionary.

If the *NanoJ program* was loaded on the controller, it is automatically executed after the controller is switched on or restarted, as long as you do not set bit 0 in object 2300_h to "0".

9.1.1 Available computing time

A *NanoJ program* receives computing time cyclically in a 1 ms clock (see following figure). Because computing time is lost through interrupts and system functions of the firmware, only a portion of the computing time is available to the user program (depending on control mode and application). In this time, the user program must run through the cycle and either complete the cycle or yield the computing time by calling the `yield()` function. In the former case, the user program is restarted with the start of the next 1 ms cycle; the latter results in the program being continued on the next 1 ms cycle with the command that follows the `yield()` function.



If the *NanoJ program* needs more time than was allotted, it is ended and an error code set in the object dictionary.

TIP
When developing user programs, the runtime behavior must be carefully examined, especially for more time-intensive tasks. For example, it is therefore recommended that tables be used instead of calculating a sine value using a `sin` function.

NOTICE

If the *NanoJ program* does not yield the computing time after too long a time, it is ended by the operating system. In this case, the number `4` is entered in the statusword for object 2301_h; in the error register for object 2302_h, the number `5` (timeout) is noted, see 2301h NanoJ Status and 2302h NanoJ Error Code.

To keep the *NanoJ program* from stopping, you can activate *AutoYield* mode by writing value "5" in 2300_h. In *AutoYield* mode, however, the *NanoJ program* is no longer real-time capable and no longer runs every 1 ms.

9.1.2 Protected runtime environment

Using process-specific properties, a so-called *protected runtime environment* is generated. A user program in the protected runtime environment is only able to access specially allocated memory areas and system resources. For example, an attempt to directly write to a processor IO register is acknowledged with an *MPU Fault* and the user program terminated with the corresponding error code in the object dictionary.

9.1.3 NanoJ program – communication possibilities

A *NanoJ program* has a number of possibilities for communicating with the controller:

- Read and write OD values using PDO mapping
- Directly read and write OD values via NanoJ functions
- Call other NanoJ functions (e.g., write debug output)

The OD values of the user program are made available in the form of variables via *PDO mapping*. Before a user program receives the 1 ms time slot, the firmware transfers the values from the object dictionary to the variables of the user program. As soon as the user program receives computing time, it can manipulate these variables as regular C variables. At the end of the time slot, the new values are then automatically copied by the firmware back to the respective OD entries.

To optimize the performance, three types of mapping are defined: input, output, and input/output (In, Out, InOut).

- *Input mappings* can only be read; they are not transferred back to the object dictionary.
- *Output mappings* can only be written.
- *Input/output mappings*, on the other hand, can both be read and written.

The set mappings can be read and checked via the GUI for objects 2310_h, 2320_h, and 2330_h. Up to 16 entries are allowed for each mapping.

Whether a variable is stored in the input, output or data range is controlled in *Plug & Drive Studio* via the specification of the *linker section*.

NanoJ inputs and NanoJ outputs

To communicate with the *NanoJ program* via the respective interface, you can use the following objects:

- 2400h NanoJ Inputs: Array with thirty-two S32 values for passing values to the *NanoJ program*
- 2410h NanoJ Init Parameters: Array with thirty-two S32 values. This object can be stored, unlike 2400_h.
- 2500h NanoJ Outputs: Array with thirty-two S32 values, where the *NanoJ program* can store values that can be read out via the fieldbus

9.1.4 Executing a NanoJ program

When executing a cycle, the *NanoJ program* essentially consists of the following three steps with respect to the PDO mapping:

1. Read values from the object dictionary and copy them to the input and output areas
2. Execute a user program
3. Copy values from the output and input areas back to the object dictionary

The configuration of the copy processes is based on the CANopen standard.

In addition, values of the object dictionary can be accessed via NanoJ functions. This is generally slower; mappings are therefore to be preferred. The number of mappings is limited (16 entries each in In/Out/InOut).

TIP

Nanotec recommends: Map OD entries that are used and changed frequently and use NanoJ function to access OD entries that are used less frequently.

A list of available NanoJ functions can be found in chapter NanoJ functions in the NanoJ program.

TIP

Nanotec recommends accessing a given OD value either by mapping or using a NanoJ function with `od_write()`. If both are used simultaneously, the NanoJ function has no effect.

9.1.5 NanoJ program – OD entries

The *NanoJ program* is controlled and configured in object range 2300_h to 2330_h (see 2300h NanoJ Control).

OD-Index	Name and description
2300_h	2300h NanoJ Control
2301_h	2301h NanoJ Status
2302_h	2302h NanoJ Error Code
2310_h	2310h NanoJ Input Data Selection
2320_h	2320h NanoJ Output Data Selection
2330_h	2330h NanoJ In/output Data Selection

Example:

To start the *TEST1.USR* user program, the following sequence can, for example, be used:

- Check entry 2302_h for error code.
- If no error:
  Start the *NanoJ program* by writing object 2300_h, bit 0 = "1" or by restarting the controller.

NOTICE

It can take up to 200 ms for the NanoJ program to start.

- Check entry 2302_h for error code and object 2301_h, bit 0 = "1".

To stop a running program: write entry 2300_h with bit 0 value = "0".

9.1.6 Structure of a NanoJ program

A user program consists of at least two instructions:

- the preprocessor instruction `#include "wrapper.h"`
- the `void user(){}` function

The code to be executed can be stored in the `void user()` function.

NOTICE

The file names of the user programs must not be longer than eight characters plus three characters in the suffix; file name `main.cpp` is permissible, file name `aLongFileName.cpp` is not permissible.

NOTICE

In *NanoJ programs*, global variables may only be initialized within functions. It then follows:

- No `new` operator
- No constructors
- No initialization of global variables outside of functions

Examples:

The global variable is to be initialized within the `void user()` function:

```
unsigned int i;
void user(){
 i = 1;
 i += 1;
}
```

The following assignment results in an error during compilation:

```
unsigned int i = 1;
 void user() {
 i += 1;
}
```

9.1.7 NanoJ program example

The example shows the programming of a square wave signal in object 2500_h:01_h.

```
// file main.cpp
map S32 outputReg1 as inout 0x2500:1
#include "wrapper.h"

// user program
void user()
{
  U16 counter = 0;
  while( 1 )
  {
    ++counter;

    if( counter < 100 )
     InOut.outputReg1 = 0;
    else if( counter < 200 )
      InOut.outputReg1 = 1;
    else
      counter = 0;

    // yield() 5 times (delay 5ms)
    for(U08 i = 0; i < 5; ++i )
      yield();
  }
}// eof
```

You can find other examples at us.nanotec.com.

9.2 Mapping in the NanoJ program

With this method, a variable in the *NanoJ program* is linked directly with an entry in the object dictionary. The creation of the mapping must be located at the start of the file here, even before the `#include "wrapper.h"` instruction.

TIP

Nanotec recommends:

- Use mapping if you need to access an object in the object dictionary frequently, e. g., *controlword* 6040_h or *statusword* 6041_h.
- The `od_write()` and `od_read()` functions are better suited for accessing objects a single time, see Accessing the object dictionary.

9.2.1 Declaration of the mapping

The declaration of the mapping is structured as follows:

```
map <TYPE> <NAME> as <input|output|inout> <INDEX>:<SUBINDEX>
```

Where:

- ```
  <TYPE>
  ```
  The data type of the variable; U32, U16, U08, S32, S16 or S08.
- `<NAME>`
  The name of the variable as it is used in the user program.
- ```
  <input|output|inout>
  ```
  The read and write permission of a variable: a variable can be declared as an `input`, `output` or `inout`. This defines whether a variable is readable (`input`), writable (`output`) or both (`inout`) and the structure by means of which it must be addressed in the program.
- ```
  <INDEX>:<SUBINDEX>
  ```
  Index and subindex of the object to be mapped in the object dictionary.

Each declared variable is addressed in the user program via one of the three structures: *In*, *Out* or *InOut* depending on the defined write and read direction.

NOTICE

A comment is only permitted above the respective mapping declaration in the code, not on the same line.

9.2.2 Example of mapping

Example of a mapping and the corresponding variable accesses:

```
// 6040h:00h is UNSIGNED16
map U16 controlWord as output 0x6040:00
// 6041h:00h is UNSIGNED16
map U16 statusWord as input 0x6041:00

// 6060h:00h is SIGNED08 (INTEGER8)
map S08 modeOfOperation as inout 0x6060:00
```
```

```
#include "wrapper.h"

void user()
{
  [...]
  Out.controlWord = 1;
  U16 tmpVar = In.statusword;
  InOut.modeOfOperation = tmpVar;
  [...]
}
```

9.2.3 Possible error at `od_write()`

A possible source of errors is a write access with the `od_write()` function (see NanoJ functions in the NanoJ program) of an object in the object dictionary that was simultaneously created as mapping. The code listed in the following is incorrect:

```
map U16 controlWord as output 0x6040:00
#include " wrapper.h"
void user()
{
 [...]
  Out.controlWord = 1;
  [...]
  od_write(0x6040, 0x00, 5 ); // der Wert wird durch das Mapping überschrieben
  [...]
}
```

The line with the `od_write(0x6040, 0x00, 5 );` command has no effect. As described in the introduction, all mappings are copied to the object dictionary at the end of each millisecond.

This results in the following sequence:

1. The `od_write` function writes the value `5` in object 6040_h:00_h.
2. At the end of the 1 ms cycle, the mapping is written that also specifies object 6040_h:00_h, however, with the value `1`.
3. From the perspective of the user, the `od_write` command thus serves no purpose.

9.3 NanoJ functions in the NanoJ program

With NanoJ functions, it is possible to call up functions integrated in the firmware directly from a user program. Code can only be directly executed in the protected area of the protected execution environment and is realized via so-called *Cortex Supervisor Calls* (Svc Calls). Here, an interrupt is triggered when the function is called, thereby giving the firmware the possibility to temporarily permit code execution outside of the protected execution environment. Developers of user programs do not need to worry about this mechanism – for them, the NanoJ functions can be called up like normal C functions. Only the *wrapper.h* file needs to be integrated as usual.

9.3.1 Accessing the object dictionary

void **od_write** (U32 index, U32 subindex, U32 value)

This function writes the transferred value to the specified location in the object dictionary.

index	Index of the object to be written in the object dictionary
subindex	Subindex of the object to be written in the object dictionary
value	Value to be written

NOTICE

It is highly recommended that the processor time be passed on with `yield()` after calling a `od_write()`. The value is immediately written to the OD. For the firmware to be able to trigger actions that are dependent on this, however, it must receive computing time. This, in turn, means that the user program must either be ended or interrupted with `yield()`.

U32 **od_read** (U32 index, U32 subindex)

This function reads the value at the specified location in the object dictionary and returns it.

index	Index of the object to be read in the object dictionary
subindex	Subindex of the object to be read in the object dictionary
Output value	Content of the OD entry

NOTICE

Active waiting for a value in the object dictionary should always be associated with a `yield()`.

Example

```
while (od_read(2400,2) != 0) // wait until 2400:2 is set
{ yield(); }
```

9.3.2 Process control

```
void yield()
```

This function returns the processor time to the operating system. In the next time slot, the program continues at the location after the call.

```
void sleep (U32 ms)
```

This function returns the processor time to the operating system for the specified number of milliseconds. The user program is then continued at the location after the call.

ms	Time to be waited in milliseconds

9.3.3 Debug output

The following functions output a value in the debug console. They differ with respect to the data type of the parameter to be passed.

```
bool VmmDebugOutputString (const char *outstring)
```

```
bool VmmDebugOutputInt (const U32 val)
```

```
bool VmmDebugOutputByte (const U08 val)
```

```
bool VmmDebugOutputHalfWord (const U16 val)
```

```
bool VmmDebugOutputWord (const U32 val)
```

```
bool VmmDebugOutputFloat (const float val)
```

NOTICE

The debug outputs are first written to a separate area of the object dictionary and read from there by the *Plug & Drive Studio*.

This OD entry has index 2600_h and is 64 characters long, see 2600h NanoJ Debug Output. Subindex 00 always contains the number of characters already written.

If the buffer is full, `VmmDebugOutputxxx()` initially fails; execution of the user program ceases and it stops at the location of the debug output. Only after the GUI has read the buffer and after subindex 00 has been reset does the program continue and `VmmDebugOutputxxx()` returns to the user program.

NOTICE

Debug outputs may therefore only be used during the test phase when developing a user program.

NOTICE

Do not use the debug output if *AutoYield* mode is activated (see Available computing time).

9.4 Restrictions and possible problems

Restrictions and possible problems when working with NanoJ are listed below:

Restriction/problem	Measure
If an object is mapped, e. g., 0x6040, the object is reset to its previous value every 1 ms. This makes it impossible to control this object via the fieldbus or the *Plug & Drive Studio*.	Instead use `od_read` / `od_write` to access the object.
If an object was mapped as output and the value of the object was never defined before starting the *NanoJ program*, the value of this object may be random.	Initialize the values of the mapped objects in your NanoJ program to ensure that it behaves deterministically.

Restriction/problem	Measure
The array initialization must not be used with more than 16 entries.	Use `constant array` instead.
Too many local variables and arrays within functions may result in a stack overflow.	Declare the variables globally. Memory requirements are monitored already during compilation; errors do not occur at runtime.
Functions that are too deeply nested may result in a stack overflow.	Observe a maximum nesting depth of 2.
`float` must not be used with comparison operators.	Use `int` instead.
`double` must not be used.	
If a NanoJ program restarts the controller (either directly with an explicit restart or indirectly, e. g., through the use of the Reset function), the controller may fall into a restart loop that can be exited only with difficulty if at all.	
`math` or `cmath` cannot be included.	

10 Description of the object dictionary

10.1 Overview

This chapter contains a description of all objects.

You will find information here on:

- Functions
- Object descriptions ("Index")
- Value descriptions ("Subindices")
- Descriptions of bits
- Description of the object

10.2 Structure of the object description

The description of the object entries always has the same structure and usually consists of the following sections:

Function

The function of the object dictionary is briefly described in this section.

Object description

This table provides detailed information on the data type, preset values and similar. An exact description can be found in section "Object description"

Value description

This table is only available with the "Array" or "Record" data type and provides exact information about the sub-entries. A more exact description of the entries can be found in section "Value description"

Description

Here, more exact information on the individual bits of an entry is provided or any compositions explained. A more exact description can be found in section "Description"

10.3 Object description

The object description consists of a table that contains the following entries:

Index

Designates the object index in hexadecimal notation.

Object name

The name of the object.

Object Code

The type of object. This can be one of the following entries:

- VARIABLE: In this case, the object consists of only a variable that is indexed with subindex 0.
- ARRAY: These objects always consists of a subindex 0 – which specifies the number of sub-entries – and the sub-entries themselves, beginning with index 1. The data type within an array never changes, i.e., sub-entry 1 and all subsequent entries are always of the same data type.
- RECORD: These objects always consists of a subindex 0 – which specifies the number of sub-entries – and the sub-entries themselves, beginning with index 1. Unlike an ARRAY, the data type of the sub-entries can vary. This means that, e.g., sub-entry 1 may be of a different data type than sub-entry 2.

- VISIBLE_STRING: The object describes a character string coded in ASCII. The length of the string is specified in subindex 0; the individual characters are stored beginning in subindex 1. These character strings are **not** terminated by a null character.

Data type

The size and interpretation of the object is specified here. The following notation is used for the "VARIABLE" object code:

- A distinction is made between entries that are signed; these are designated with the prefix "SIGNED". For entries that are unsigned, the prefix "UNSIGNED" is used.
- The size of the variable in bits is placed before the prefix and can be 8, 16 or 32.

Savable

Described here is whether this object is savable and, if so, in which category.

Firmware version

The firmware version beginning with which the object is available is entered here.

Change history (ChangeLog)

Any changes to the object are noted here.

There are also the following table entries for the "VARIABLE" data type:

Access

The access restriction is entered here. The following restrictions are available:

- "read/write": The object can both be read as well as written
- "read only": The object can only be read from the object dictionary. It is not possible to set a value.

PDO mapping

Some bus systems, such as CANopen or EtherCAT, support PDO mapping. Described in this table entry is whether the object can be inserted into a mapping and, if so, into which. The following designations are available here:

- "no": The object may not be entered in a mapping.
- "TX-PDO": The object may be entered in an RX mapping.
- "RX-PDO": The object may be entered in a TX mapping.

Allowed values

In some cases, only certain values may be written in the object. If this is the case, these values are listed here. If there are no restrictions, the field is empty.

Preset value

To bring the controller to a secured state when switching on, it is necessary to preset a number of objects with values. The value that is written in the object when the controller is started is noted in this table entry.

10.4 Value description

NOTICE

For the sake of clarity, a number of subindices are grouped together if the entries all have the same name.

Listed in the table with the "Value description" heading are all data for sub-entries with subindex 1 or higher. The table contains the following entries:

Subindex

Number of the currently written sub-entry.

Name

Name of the sub-entry.

Data type

The size and interpretation of the sub-entry is specified here. The following notation always applies here:

- A distinction is made between entries that are signed; these are designated with the prefix "SIGNED". For entries that are unsigned, the prefix "UNSIGNED" is used.
- The size of the variable in bits is placed before the prefix and can be 8, 16 or 32.

Access

The access restriction for the sub-entry is entered here. The following restrictions are available:

- "read/write": The object can both be read as well as written
- "read only": The object can only be read from the object dictionary. It is not possible to set a value.

PDO mapping

Some bus systems, such as CANopen or EtherCAT, support PDO mapping. Described in this table entry is whether the sub-entry can be inserted into a mapping and, if so, into which. The following designations are available here:

- "no": The object may not be entered in a mapping.
- "TX-PDO": The object may be entered in an RX mapping.
- "RX-PDO": The object may be entered in a TX mapping.

Allowed values

In some cases, only certain values may be written in the sub-entry. If this is the case, these values are listed here. If there are no restrictions, the field is empty.

Preset value

To bring the controller to a secured state when switching on, it is necessary to preset a number of sub-entries with values. The value that is written in the sub-entry when the controller is started is noted in this table entry.

10.5 Description

This section may be present if use requires additional information. If individual bits of an object or sub-entry have different meaning, diagrams as shown in the following example are used.

Example: The object is 8 bits in size; bit 0 and bit 1 have different functions. Bits 2 and 3 are grouped into one function; the same applies for bits 4 to 7.

7	6	5	4	3	2	1	0
Example [4]				Example [2]		B	A

Example [4]

Description of bit 4 up to and including bit 7; these bits are logically related. The 4 in square brackets specifies the number of related bits. A list with possible values and their description is often attached at this point.

Example [2]

Description of bits 3 and 2; these bits are logically related. The 2 in square brackets specifies the number of related bits.

- Value 00_b: The description here applies if bit 2 and bit 3 are "0".
- Value 01_b: The description here applies if bit 2 is "0" and bit 3 is "1".
- Value 10_b: The description here applies if bit 2 is "1" and bit 3 is "0".
- Value 11_b: The description here applies if bit 2 and bit 3 are "1".

B

Description of bit B; no length is specified for a single bit.

A

Description of bit A; bits with a gray background are not used.

1000h Device Type

Function

Describes the controller type.

Object description

Index	1000_h
Object name	Device Type
Object Code	VARIABLE
Data type	UNSIGNED32
Savable	no
Access	read only
PDO mapping	no
Allowed values	
Preset value	00060192_h
Firmware version	FIR-v1426
Change history	

Description

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Motor Type [16]															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Device profile number [16]															

Motor Type[16]

Describes the supported motor type. The following values are possible:

- Bit 23 to bit 16: Value "2": BLDC motor
- Bit 23 to bit 16: Value "4": Stepper motor
- Bit 23 to bit 16: Value "6": Stepper motor as well as BLDC motor

Device profile number[16]

Describes the supported CANopen standard.

Values:

0192_h or 0402_d (preset value): The CiA 402 standard is supported.

1001h Error Register

Function

Error register: In the event of an error, the corresponding error bit(s) is/are set. If the error no longer exists, it is deleted automatically.

NOTICE

For each error that occurs, a more precise error code is stored in object 1003_h.

Object description

Index	1001_h
Object name	Error Register
Object Code	VARIABLE
Data type	UNSIGNED8
Savable	no
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	00_h
Firmware version	FIR-v1426
Change history	

Description

7	6	5	4	3	2	1	0
MAN	RES	PROF	COM	TEMP	VOL	CUR	GEN

GEN

General error, always set in the event of an error

CUR

Current

VOL

Voltage

TEMP

Temperature

COM

Communication

PROF

Relates to the device profile

RES

Reserved, always "0"

MAN

Manufacturer-specific

1003h Pre-defined Error Field

Function

This object contains an error stack with up to eight entries.

Object description

Index	1003_h
Object name	Pre-defined Error Field
Object Code	ARRAY
Data type	UNSIGNED32
Savable	no
Firmware version	FIR-v1426
Change history	

Value description

Subindex	00_h
Name	Number Of Errors
Data type	UNSIGNED8
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00_h

Subindex	01_h
Name	1st Standard Error Field
Data type	UNSIGNED32
Access	read only
PDO mapping	no
Allowed values	
Preset value	00000000_h

Subindex	02_h
Name	2nd Standard Error Field
Data type	UNSIGNED32
Access	read only
PDO mapping	no
Allowed values	

Preset value	00000000_h

Subindex	03_h
Name	3th Standard Error Field
Data type	UNSIGNED32
Access	read only
PDO mapping	no
Allowed values	
Preset value	00000000_h

Subindex	04_h
Name	4th Standard Error Field
Data type	UNSIGNED32
Access	read only
PDO mapping	no
Allowed values	
Preset value	00000000_h

Subindex	05_h
Name	5th Standard Error Field
Data type	UNSIGNED32
Access	read only
PDO mapping	no
Allowed values	
Preset value	00000000_h

Subindex	06_h
Name	6th Standard Error Field
Data type	UNSIGNED32
Access	read only
PDO mapping	no
Allowed values	
Preset value	00000000_h

Subindex	07_h
Name	7th Standard Error Field
Data type	UNSIGNED32
Access	read only
PDO mapping	no
Allowed values	
Preset value	00000000_h

Subindex	08_h

Name	8th Standard Error Field
Data type	UNSIGNED32
Access	read only
PDO mapping	no
Allowed values	
Preset value	00000000_h

Description

General function

If a new error occurs, it is entered in subindex 1. The already existing entries in subindices 1 to 7 are moved back one position. The error in subindex 7 is thereby removed.

The number of errors that have already occurred can be read from the object with subindex 0. If no error is currently entered in the error stack, it is not possible to read one of the eight subindices 1–8 and an error (abort code = 08000024_h) is sent in response. If a "0" is written in subindex 0, counting starts again from the beginning.

Bit description

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Error Number [8]								Error Class [8]							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Error Code [16]															

Error Number [8]

This can be used to pinpoint the cause of the error. The meaning of the number can be found in the following table.

Error number	Description
0	Watchdog-Reset
1	Input voltage (+Ub) too high
2	Output current too high
3	Input voltage (+Ub) too low
4	Error at fieldbus
6	CANopen only: NMT master takes too long to send Nodeguarding request
7	Sensor 1 (see 3204_h): Error through electrical fault or defective hardware
8	Sensor 2 (see 3204_h): Error through electrical fault or defective hardware
9	Sensor 3 (see 3204_h): Error through electrical fault or defective hardware
10	Warning: Positive limit switch exceeded
11	Warning: Negative limit switch exceeded
12	Overtemperature error
13	The values of object 6065_h (Following Error Window) and object 6066_h (Following Error Time Out) were exceeded; a fault was triggered.
14	Warning: Nonvolatile memory full. The current save process could not be completed; parts of the data of the save process are lost. Controller must be restarted for cleanup work.
15	Motor blocked
16	Warning: Nonvolatile memory damaged; controller must be restarted for cleanup work (all saved objects are reset to default).
17	CANopen only: Slave took too long to send PDO messages.

Error number	Description
18	Sensor n (see 3204_h), where n is greater than 3: Error through electrical fault or defective hardware
19	CANopen only: PDO not processed due to a length error
20	CANopen only: PDO length exceeded
21	Warning: Restart the controller to avoid future errors when saving (nonvolatile memory full/corrupt).
22	Rated current must be set ($203B_h$:01_h/6075_h)
23	Encoder resolution, number of pole pairs and some other values are incorrect.
24	Motor current is too high, adjust the PI parameters.
25	Internal software error, generic
26	Current too high at digital output
27	CANopen only: Unexpected sync length
30	Error in speed monitoring: slippage error too large
32	Internal error: Correction factor for reference voltage missing in the OTP
40	Warning: Ballast resistor thermally overloaded
41	Only EtherCAT: *Sync Manager Watchdog*: The controller has not received any PDO data for an excessively long period of time; check the software and hardware connections.
46	Interlock error: Bit 3 in $60FD_h$ is set to "0", the motor may not start
48	Only CANopen: NMT status has been set to *stopped*

Error Class[8]

This byte is identical to object 1001_h

Error Code[16]

Refer to the following table for the meaning of the bytes.

Error Code	Description
1000_h	General error
2300_h	Current at the controller output too large
3100_h	Overvoltage/undervoltage at controller input
4200_h	Temperature error within the controller
5440_h	Interlock error: Bit 3 in $60FD_h$ is set to "0", the motor may not start (see the section *Interlock function* in the chapter Digital inputs)
6010_h	Software reset (watchdog)
6100_h	Internal software error, generic
6320_h	Rated current must be set ($203B_h$:01_h/6075_h)
7113_h	Warning: Ballast resistor thermally overloaded
7121_h	Motor blocked
7200_h	Internal error: Correction factor for reference voltage missing in the OTP
7305_h	Sensor 1 (see 3204_h) faulty
7306_h	Sensor 2 (see 3204_h) faulty
7307_h	Sensor n (see 3204_h), where n is greater than 2

Error Code	Description
7600h	Warning: Nonvolatile memory full or corrupt; restart the controller for cleanup work
8100h	Error during fieldbus monitoring
8130h	CANopen only: "Life Guard" error or "Heartbeat" error
8200h	CANopen only: Slave took too long to send PDO messages.
8210h	CANopen only: PDO was not processed due to a length error
8220h	CANopen only: PDO length exceeded
8240h	CANopen only: unexpected sync length
8400h	Error in speed monitoring: slippage error too large
8611h	Position monitoring error: Following error too large
8612h	Position monitoring error: Limit switch exceeded

1005h COB-ID Sync

Function

Defines the COB-ID of the SYNC message for the SYNC protocol. The value must correspond to an 11-bit-long CAN-ID and is evaluated when the controller is restarted or on a Reset Communication command.

NOTICE

If the CAN-ID is not to correspond to the default value of 80h, it must be ensured that only not-yet unassigned or reserved CAN-IDs are used.

You can activate the generation of sync messages (the controller becomes the *sync master of the network*) by setting bit 30 to "1". Set the cycle time in object 1006h.

Object description

Index	1005h
Object name	COB-ID Sync
Object Code	VARIABLE
Data type	UNSIGNED32
Savable	yes, category: communication
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000080h
Firmware version	FIR-v1426
Change history	

1006h Communication Cycle Period

Function

Contains the cycle time for the generated sync messages (see 1005_h) in µs. Only multiples of 1000 µs are permitted.

Object description

Index	1006_h
Object name	Communication Cycle Period
Object Code	VARIABLE
Data type	UNSIGNED32
Savable	yes, category: communication
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000_h
Firmware version	FIR-v2013-B726332
Change history	

1007h Synchronous Window Length

Function

This object contains the length of the time window in microseconds for synchronous PDOs. If the synchronous time window has elapsed, all synchronous TxPDOs are rejected. The RxPDOs are also rejected up to the next SYNC message.

The value "0" switches off the time window, thereby allowing the PDOs to be sent at any time.

This object is only available in device variants with CANopen connection.

Object description

Index	1007_h
Object name	Synchronous Window Length
Object Code	VARIABLE
Data type	UNSIGNED32
Savable	yes, category: communication
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000_h
Firmware version	FIR-v1426
Change history	

1008h Manufacturer Device Name

Function

Contains the device name as character string.

Object description

Index	1008_h
Object name	Manufacturer Device Name
Object Code	VARIABLE
Data type	VISIBLE_STRING
Savable	no
Access	read only
PDO mapping	no
Allowed values	
Preset value	■ PD1-C281S15-E-20-2: PD1-C281S15-E-20-2 ■ PD1-C281S15-E-20-5: PD1-C281S15-E-20-5 ■ PD1-C281S15-E-65-2: PD1-C281S15-E-65-2 ■ PD1-C281S15-E-65-5: PD1-C281S15-E-65-5 ■ PD1-C281S15-E-OF-2: PD1-C281S15-E-OF-2 ■ PD1-C281S15-E-OF-5: PD1-C281S15-E-OF-5 ■ PD1-C281L15-E-20-2: PD1-C281L15-E-20-2 ■ PD1-C281L15-E-20-5: PD1-C281L15-E-20-5 ■ PD1-C281L15-E-65-2: PD1-C281L15-E-65-2 ■ PD1-C281L15-E-65-5: PD1-C281L15-E-65-5 ■ PD1-C281L15-E-OF-2: PD1-C281L15-E-OF-2 ■ PD1-C281L15-E-OF-5: PD1-C281L15-E-OF-5
Firmware version	FIR-v1426
Change history	

1009h Manufacturer Hardware Version

Function

This object contains the hardware version as character string.

Object description

Index	1009_h
Object name	Manufacturer Hardware Version
Object Code	VARIABLE
Data type	VISIBLE_STRING
Savable	no
Access	read only
PDO mapping	no
Allowed values	
Preset value	0
Firmware version	FIR-v1426

Change history	

100Ah Manufacturer Software Version

Function

This object contains the software version as character string.

Object description

Index	$100A_h$
Object name	Manufacturer Software Version
Object Code	VARIABLE
Data type	VISIBLE_STRING
Savable	no
Access	read only
PDO mapping	no
Allowed values	
Preset value	FIR-v2425-B1060841
Firmware version	FIR-v1426
Change history	

1010h Store Parameters

Function

This object is used to start the saving of objects. See chapter Saving objects.

Object description

Index	1010_h
Object name	Store Parameters
Object Code	ARRAY
Data type	UNSIGNED32
Savable	no
Access	read only
PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v1426
Change history	Firmware version FIR-v1436: "Object name" entry changed from "Store Parameter" to "Store Parameters". Firmware version FIR-v1436: The number of entries was changed from 3 to 4. Firmware version FIR-v1512: The number of entries was changed from 4 to 5. Firmware version FIR-v1540: The number of entries was changed from 5 to 7.

	Firmware version FIR-v1738-B501312: The number of entries was changed from 7 to 14. Firmware version FIR-v2412-B1057638: entry "Name" changed from "Save Miscellaneous Configurations To Non-volatile Memory" to "Save Reserved0 Configurations To Non-volatile Memory".

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	$0D_h$

Subindex	01_h
Name	Save All Parameters To Non-volatile Memory
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000001_h

Subindex	02_h
Name	Save Communication Parameters To Non-volatile Memory
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000001_h

Subindex	03_h
Name	Save Application Parameters To Non-volatile Memory
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000001_h

Subindex	04_h
Name	Save Customer Parameters To Non-volatile Memory
Data type	UNSIGNED32
Access	read / write

PDO mapping	no
Allowed values	
Preset value	00000001_h

Subindex	05_h
Name	Save Drive Parameters To Non-volatile Memory
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000001_h

Subindex	06_h
Name	Save Tuning Parameters To Non-volatile Memory
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000001_h

Subindex	07_h
Name	Save Reserved0 Configurations To Non-volatile Memory
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000_h

Subindex	08_h
Name	Save Reserved1 Configurations To Non-volatile Memory
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000_h

Subindex	09_h
Name	Save Reserved2 Configurations To Non-volatile Memory
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000_h

Subindex	$0A_h$
Name	Save CANopen Configurations To Non-volatile Memory
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000001_h

Subindex	$0B_h$
Name	Save Modbus RTU Configurations To Non-volatile Memory
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000001_h

Subindex	$0C_h$
Name	Save Ethernet Configurations To Non-volatile Memory
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000001_h

Subindex	$0D_h$
Name	Save Profibus Configurations To Non-volatile Memory
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000001_h

Description

Each subindex of the object stands for a certain memory class. By reading out the entry, it is possible to determine whether (value "1") or not (value="0") this memory category can be saved.

To start the save process of a memory category, value "65766173_h" must be written in the corresponding subindex. This corresponds to the decimal of 1702257011_d or the ASCII string `save`. As soon as the saving process is completed, the save command is again overwritten with the value "1", since saving is possible again.

For a detailed description, see chapter Saving objects.

1011h Restore Default Parameters

Function

This object can be used to reset all or part of the object dictionary to the default values. See chapter Saving objects.

Object description

Index	1011_h
Object name	Restore Default Parameters
Object Code	ARRAY
Data type	UNSIGNED32
Savable	no
Access	read only
PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v1426
Change history	Firmware version FIR-v1436: "Object Name" entry changed from "Restore Default Parameter" to "Restore Default Parameters". Firmware version FIR-v1436: The number of entries was changed from 2 to 4. Firmware version FIR-v1512: The number of entries was changed from 4 to 5. Firmware version FIR-v1512: "Name" entry changed from "Restore The Comm Default Parameters" to "Restore Communication Default Parameters". Firmware version FIR-v1512: "Name" entry changed from "Restore The Application Default Parameters" to "Restore Application Default Parameters". Firmware version FIR-v1540: The number of entries was changed from 5 to 7. Firmware version FIR-v1738-B501312: The number of entries was changed from 7 to 14. Firmware version FIR-v2412-B1057638: entry "Name" changed from "Restore Miscellaneous Configurations" to "Restore Reserved0 Configurations To Non-volatile Memory".

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	$0D_h$

Subindex	01_h
Name	Restore All Default Parameters
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000001_h

Subindex	02_h
Name	Restore Communication Default Parameters
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000001_h

Subindex	03_h
Name	Restore Application Default Parameters
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000001_h

Subindex	04_h
Name	Restore Customer Default Parameters
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000001_h

Subindex	05_h
Name	Restore Drive Default Parameters
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000001_h

Subindex	06_h
Name	Restore Tuning Default Parameters
Data type	UNSIGNED32

Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000001_h

Subindex	07_h
Name	Restore Reserved0 Configurations To Non-volatile Memory
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000_h

Subindex	08_h
Name	Restore Reserved1 Configurations To Non-volatile Memory
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000_h

Subindex	09_h
Name	Restore Reserved2 Configurations To Non-volatile Memory
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000_h

Subindex	$0A_h$
Name	Restore CANopen Configurations To Non-volatile Memory
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000001_h

Subindex	$0B_h$
Name	Restore Modbus RTU Configurations To Non-volatile Memory
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	

Preset value	00000001_h

Subindex	$0C_h$
Name	Restore Ethernet Configurations To Non-volatile Memory
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000001_h

Subindex	$0D_h$
Name	Restore Profibus Configurations To Non-volatile Memory
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000001_h

Description

If the value $64616F6C_h$ (or 1684107116_d or ASCII `load`) is written in this object, part or all of the object dictionary is reset to the default values. The subindex that is used decides which range is reset.

For a detailed description, see chapter Discarding the saved data.

1014h COB-ID EMCY

Function

This object describes the COB-ID of the "Emergency Service" under CANopen.

With the *Valid Bit* (bit 31) = "1", the Emergency Service can be deactivated; the service is active with the value "0". Every time the controller is restarted, bits 0 to 30 are generated according to the node-ID.

Object description

Index	1014_h
Object name	COB-ID EMCY
Object Code	VARIABLE
Data type	UNSIGNED32
Savable	yes, category: communication
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000_h
Firmware version	FIR-v1426
Change history	Firmware version FIR-v1540: "Access" table entry for subindex 00 changed from "read only" to "read/write".

1018h Identity Object

Function

This object returns general information on the device, such as manufacturer, product code, revision and serial number.

TIP
Have these values ready in the event of service inquiries.

Object description

Index	1018_h
Object name	Identity Object
Object Code	RECORD
Data type	IDENTITY
Savable	no
Firmware version	FIR-v1426
Change history	

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	04_h

Subindex	01_h
Name	Vendor ID
Data type	UNSIGNED32
Access	read only
PDO mapping	no
Allowed values	
Preset value	$0000026C_h$

Subindex	02_h
Name	Product Code
Data type	UNSIGNED32
Access	read only
PDO mapping	no
Allowed values	
Preset value	■ PD1-C281S15-E-20-2: 00210100_h

- PD1-C281S15-E-20-5: 00210102_h
- PD1-C281S15-E-65-2: 00210200_h
- PD1-C281S15-E-65-5: 00210202_h
- PD1-C281S15-E-OF-2: 00210300_h
- PD1-C281S15-E-OF-5: 00210302_h
- PD1-C281L15-E-20-2: 00230100_h
- PD1-C281L15-E-20-5: 00230102_h
- PD1-C281L15-E-65-2: 00230200_h
- PD1-C281L15-E-65-5: 00230202_h
- PD1-C281L15-E-OF-2: 00230300_h
- PD1-C281L15-E-OF-5: 00230302_h

Subindex	03_h
Name	Revision Number
Data type	UNSIGNED32
Access	read only
PDO mapping	no
Allowed values	
Preset value	09790000_h

Subindex	04_h
Name	Serial Number
Data type	UNSIGNED32
Access	read only
PDO mapping	no
Allowed values	
Preset value	00000000_h

1020h Verify Configuration

Function

This object indicates the date and time that the configuration was stored.

A configuration tool or a master can use this object to verify the configuration after a reset and, if necessary, perform a new configuration.

The tool must set the date and time before the storage mechanism is started (see chapter Saving objects).

Object description

Index	1020_h
Object name	Verify Configuration
Object Code	ARRAY
Data type	UNSIGNED32
Savable	yes, category: verify
Access	read only
PDO mapping	no
Allowed values	

Preset value	
Firmware version	FIR-v1540
Change history	

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	02_h

Subindex	01_h
Name	Configuration Date
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000_h

Subindex	02_h
Name	Configuration Time
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000_h

Description

Subindex 01_h (configuration date) is to contain the number of days since 1 January 1984.

Subindex 02_h (configuration time) is to contain the number of milliseconds since midnight.

1F50h Program Data

Function

This object is used to program memory areas of the controller. Each entry stands for a certain memory area.

Object description

Index	$1F50_h$
Object name	Program Data
Object Code	ARRAY

Data type	DOMAIN
Savable	no
Access	read only
PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v1540
Change history	

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	02_h

Subindex	01_h
Name	Program Data Bootloader/firmware
Data type	DOMAIN
Access	read / write
PDO mapping	no
Allowed values	
Preset value	0

Subindex	02_h
Name	Program Data NanoJ
Data type	DOMAIN
Access	read / write
PDO mapping	no
Allowed values	
Preset value	0

1F51h Program Control

Function

This object is used to control the programming of memory areas of the controller. Each entry stands for a certain memory area.

Object description

Index	$1F51_h$
Object name	Program Control

Object Code	ARRAY
Data type	UNSIGNED8
Savable	no
Access	read only
PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v1540
Change history	

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	02_h

Subindex	01_h
Name	Program Control Bootloader/firmware
Data type	UNSIGNED8
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00_h

Subindex	02_h
Name	Program Control NanoJ
Data type	UNSIGNED8
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00_h

1F57h Program Status

Function

This object indicates the programming status during the programming of memory areas of the controller. Each entry stands for a certain memory area.

Object description

Index	$1F57_h$

Object name	Program Status
Object Code	ARRAY
Data type	UNSIGNED32
Savable	no
Access	read only
PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v1540
Change history	

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	02_h

Subindex	01_h
Name	Program Status Bootloader/firmware
Data type	UNSIGNED32
Access	read only
PDO mapping	no
Allowed values	
Preset value	00000000_h

Subindex	02_h
Name	Program Status NanoJ
Data type	UNSIGNED32
Access	read only
PDO mapping	no
Allowed values	
Preset value	00000000_h

2028h MODBUS Slave Address

Function

This object contains the slave address for Modbus.

Object description

Index	2028_h
Object name	MODBUS Slave Address
Object Code	VARIABLE
Data type	UNSIGNED8
Savable	yes, category: Modbus RTU
Access	read / write
PDO mapping	no
Allowed values	1-247
Preset value	05_h
Firmware version	FIR-v1436
Change history	Firmware version FIR-v1748-B531667: "Savable" entry changed from "yes, category: communication" to "yes, category: Modbus RTU".

202Ah MODBUS RTU Baudrate

Function

This object contains the baud rate of the Modbus in Bd.

Object description

Index	$202A_h$
Object name	MODBUS RTU Baudrate
Object Code	VARIABLE
Data type	UNSIGNED32
Savable	yes, category: Modbus RTU
Access	read / write
PDO mapping	no
Allowed values	
Preset value	$00004B00_h$
Firmware version	FIR-v1436
Change history	Firmware version FIR-v1748-B531667: "Savable" entry changed from "yes, category: communication" to "yes, category: Modbus RTU".

202Ch MODBUS RTU Stop Bits

Function

This object contains the number of stop bits of the Modbus.

Object description

Index	$202C_h$
Object name	MODBUS RTU Stop Bits
Object Code	VARIABLE

Data type	UNSIGNED8
Savable	no
Access	read only
PDO mapping	no
Allowed values	
Preset value	01_h
Firmware version	FIR-v1436
Change history	Firmware version FIR-v1540: "Savable" entry changed from "yes, category: communication" to "no". Firmware version FIR-v1540: "Access" table entry for subindex 00 changed from "read/write" to "read only".

Description

The number of stop bits is dependent on the parity, which can be set in object $202D_h$.

202Dh MODBUS RTU Parity

Function

For Modbus RTU, this object sets the number of parity bits and stop bits.

Object description

Index	$202D_h$
Object name	MODBUS RTU Parity
Object Code	VARIABLE
Data type	UNSIGNED8
Savable	yes, category: Modbus RTU
Access	read / write
PDO mapping	no
Allowed values	
Preset value	04_h
Firmware version	FIR-v1540
Change history	Firmware version FIR-v1748-B531667: "Savable" entry changed from "yes, category: communication" to "yes, category: Modbus RTU".

Description

The following values apply:

- Value "0x00": Parity None, Stop Bits 2
- Value "0x04": Parity Even, Stop Bits 1
- Value "0x06": Parity Odd, Stop Bits 1

2030h Pole Pair Count

Function

Contains the number of pole pairs of the connected motor.

Object description

Index	2030_h
Object name	Pole Pair Count
Object Code	VARIABLE
Data type	UNSIGNED16
Savable	yes, category: tuning
Access	read / write
PDO mapping	no
Allowed values	
Preset value	0032_h
Firmware version	FIR-v1426
Change history	Firmware version FIR-v1540: "Savable" entry changed from "no" to "yes, category: tuning". Firmware version FIR-v2315-B1040535: "Data type" entry changed from "UNSIGNED32" to "UNSIGNED16". Firmware version FIR-v2239-B1032745: "Datentyp" entry changed from "UNSIGNED16" to "UNSIGNED32". Firmware version FIR-v2339-B1048823: "Data type" entry changed from "UNSIGNED32" to "UNSIGNED16".

2031h Max Motor Current

Function

Enter the maximum permissible motor current in milliamperes here. All current values are limited by this value.

Within the controller, the entered value is always interpreted as the root mean square.

Object description

Index	2031_h
Object name	Max Motor Current
Object Code	VARIABLE
Data type	UNSIGNED32
Savable	yes, category: tuning
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000_h
Firmware version	FIR-v1426
Change history	Firmware version FIR-v1614: "Savable" entry changed from "yes, category: application" to "yes, category: tuning". Firmware version FIR-v1614: "Object Name" entry changed from "Peak Current" to "Max Current". Firmware version FIR-v1748-B538662: "Object Name" entry changed from "Maximum Current" to "Max Motor Current".

	Firmware version FIR-v1825-B577172: "Object Name" entry changed from "Max Motor Current" to "Maximum Current".
	Firmware version FIR-v1825-B577172: "Object Name" entry changed from "Maximum Current" to "Max Motor Current".
	Firmware version FIR-v1825-B577172: "Object Name" entry changed from "Max Motor Current" to "Maximum Current".
	Firmware version FIR-v1825-B577172: "Object Name" entry changed from "Maximum Current" to "Max Motor Current".

2034h Upper Voltage Warning Level

Function

This object contains the threshold value for the "overvoltage" error in millivolts.

Object description

Index	2034_h
Object name	Upper Voltage Warning Level
Object Code	VARIABLE
Data type	UNSIGNED32
Savable	yes, category: application
Access	read / write
PDO mapping	no
Allowed values	
Preset value	$00007D00_h$
Firmware version	FIR-v1426
Change history	

Description

If the input voltage of the controller exceeds this threshold value, the motor is switched off and an error triggered. This error is reset automatically if the input voltage is less than (voltage of object 2034_h minus 2 volts).

2035h Lower Voltage Warning Level

Function

This object contains the threshold value for the "Undervoltage" error in millivolts.

Object description

Index	2035_h
Object name	Lower Voltage Warning Level
Object Code	VARIABLE
Data type	UNSIGNED32
Savable	yes, category: application
Access	read / write
PDO mapping	no

Allowed values	
Preset value	00002904_h
Firmware version	FIR-v1426
Change history	

Description

If the input voltage of the controller falls below this threshold value, the motor is switched off and an error triggered. The error is reset automatically if the input voltage exceeds the voltage of object 2035_h plus 1.5 volts.

2036h Open Loop Current Reduction Idle Time

Function

This object describes the time in milliseconds that the motor must be at a standstill before current reduction is activated.

Object description

Index	2036_h
Object name	Open Loop Current Reduction Idle Time
Object Code	VARIABLE
Data type	UNSIGNED32
Savable	yes, category: application
Access	read / write
PDO mapping	no
Allowed values	
Preset value	$000003E8_h$
Firmware version	FIR-v1426
Change history	

2037h Open Loop Current Reduction Value/factor

Function

This object describes the rms current to which the motor current in open-loop is to be reduced if the motor is at a standstill.

Object description

Index	2037_h
Object name	Open Loop Current Reduction Value/factor
Object Code	VARIABLE
Data type	INTEGER32
Savable	yes, category: application
Access	read / write
PDO mapping	no
Allowed values	
Preset value	$FFFFFFCE_h$

Firmware version	FIR-v1426
Change history	

Description

Value of 2037$_h$ greater than or equal to 0 and less than value 6075$_h$

Current is reduced to the value entered here. The value is in mA and interpreted as root mean square.

Value of 2037$_h$ in the range from -1 to -100

The entered value is interpreted as a percentage and determines the reduction of the rated current in 2037$_h$. The value in 6075$_h$ is used for the calculation.

Example: Object 6075$_h$ has the value 4200 mA. The value -60 in 2037$_h$ reduces the current by 60% of 6075$_h$. The result is a current reduction to a root mean square of 6075$_h$ * (2037$_h$ + 100) / 100 = 1680 mA.

The value -100 in 2037$_h$ would, for example, mean that a current reduction is set to a root mean square of 0 mA.

2038h Brake Controller Timing

Function

This object contains the times for the *brake control* in milliseconds as well as the PWM frequency and the duty cycle.

Object description

Index	2038$_h$
Object name	Brake Controller Timing
Object Code	ARRAY
Data type	UNSIGNED32
Savable	yes, category: application
Firmware version	FIR-v1426
Change history	

Value description

Subindex	00$_h$
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	06$_h$

Subindex	01$_h$
Name	Close Brake Idle Time
Data type	UNSIGNED32

Access	read / write
PDO mapping	no
Allowed values	
Preset value	$000003E8_h$

Subindex	02_h
Name	Shutdown Power Idle Time
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	$000003E8_h$

Subindex	03_h
Name	Open Brake Delay Time
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	$000003E8_h$

Subindex	04_h
Name	Start Operation Delay Time
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000_h

Subindex	05_h
Name	PWM Frequency
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	between 50 and
Preset value	$00004E20_h$

Subindex	06_h
Name	PWM Duty Cycle
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	0, between 2 and 100 (64_h)

Preset value	00000000_h

Description

The subindices have the following functions:

- 01_h: Time between motor standstill and the closing of the brake.
- 02_h: Time between the closing of the brake and the switching off of the motor current.
- 03_h: Time between the switching on of the motor current and opening of the brake.
- 04_h: Time between the opening of the brake and when the *Operation enabled* state of the CiA 402 Power State Machine is reached.
- 05_h: Frequency of the PWM signal in hertz.
- 06_h: Duty cycle of the PWM signal in percent.

2039h Motor Currents

Function

This object contains the measured motor currents in mA. All values are peak values, (#2*rms).

Object description

Index	2039_h
Object name	Motor Currents
Object Code	ARRAY
Data type	INTEGER32
Savable	no
Access	read only
PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v1426
Change history	Firmware version FIR-v1504: "PDO mapping" table entry for subindex 01 changed from "no" to "TX-PDO". Firmware version FIR-v1504: "PDO mapping" table entry for subindex 02 changed from "no" to "TX-PDO". Firmware version FIR-v1504: "PDO mapping" table entry for subindex 03 changed from "no" to "TX-PDO". Firmware version FIR-v1504: "PDO mapping" table entry for subindex 04 changed from "no" to "TX-PDO". Firmware version FIR-v2213: subindex 05_h, "Actual Current" added. Phase currents Ia and Ib changed to Iα and Iβ (Clarke transformation).

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only

PDO mapping	no
Allowed values	
Preset value	05_h

Subindex	01_h
Name	Id
Data type	INTEGER32
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	00000000_h

Subindex	02_h
Name	Iq
Data type	INTEGER32
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	00000000_h

Subindex	03_h
Name	Iα
Data type	INTEGER32
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	00000000_h

Subindex	04_h
Name	Iβ
Data type	INTEGER32
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	00000000_h

Subindex	05_h
Name	Actual Current
Data type	INTEGER32
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	00000000_h

Description

- 01_h: Field-forming components of the current
- 02_h: Torque-forming components of the current
- 03_h: Iα
- 04_h: Iβ
- 05_h: total current divided by √2, i.e., calculated down to a motor phase. In *closed-loop*, the sign of Iq is also used. The current value can then be placed on a scale to compare with the current from 6075_h, 2031_h and $203B_h$:05_h.
  open-loop: $I = \sqrt{(I\alpha^2 + I\beta^2)} / \sqrt{2}$
  Closed Loop: $I = \mathrm{sgn}(Iq) * \sqrt{(I\alpha^2 + I\beta^2)} / \sqrt{2}$

NOTICE

Motor currents I_d (subindex 01_h) and I_q (subindex 02_h) are only displayed if closed-loop was activated; the value 0 is otherwise output.

203Ah Homing On Block Configuration

Function

This object contains the parameters for *Homing on Block* (see chapter Homing).

Object description

Index	$203A_h$
Object name	Homing On Block Configuration
Object Code	ARRAY
Data type	INTEGER32
Savable	yes, category: application
Access	
PDO mapping	
Allowed values	
Preset value	
Firmware version	FIR-v1426
Change history	Firmware version FIR-v1540: The number of entries was changed from 4 to 3. Firmware version FIR-v1540: "Name" entry changed from "Period Of Blocking" to "Block Detection time". Firmware version FIR-v1614: "Data Type" entry changed from "UNSIGNED32" to "INTEGER32". Firmware version FIR-v1614: "Savable" entry changed from "no" to "yes, category: application". Firmware version FIR-v1614: "Data Type" entry changed from "UNSIGNED32" to "INTEGER32". Firmware version FIR-v1614: "Data Type" entry changed from "UNSIGNED32" to "INTEGER32".

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	02_h

Subindex	01_h
Name	Minimum Current For Block Detection
Data type	INTEGER32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	$FFFFFFBA_h$

Subindex	02_h
Name	Block Detection Time
Data type	INTEGER32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	$000000C8_h$

Description

The subindices have the following function:

- 01_h: Specifies the current limit value above which blocking is to be detected. Positive numerical values specify the current limit in mA, negative numbers specify a percentage of object 2031_h. Example: The value "1000" corresponds to 1000 mA (= 1 A); the value "-70" corresponds to 70% of 2031_h.
- 02_h: Specifies the time in ms that the motor is to continue to travel against the block after block detection.

203Bh I2t Parameters

Function

This object contains the parameters for I^2t monitoring.

I^2t monitoring is activated by entering a value greater than 0 in $203B_h$:01 and $203B_h$:02 and a value greater than 1000 in 6073_h (see I2t Motor overload protection).

With one exception, I^2t monitoring can only be used for *closed-loop* mode: If I^2t is activated in *open-loop* mode, the current is reduced to the smaller of $203B_h$:01_h, 6073_h and 2031_h.

Object description

Index	$203B_h$
Object name	I2t Parameters

Object Code	ARRAY
Data type	UNSIGNED32
Savable	yes, category: tuning
Firmware version	FIR-v1426
Change history	Firmware version FIR-v1512: "Savable" entry changed from "no" to "yes, category: application". Firmware version FIR-v1512: The number of entries was changed from 7 to 8. Firmware version FIR-v1614: "Savable" entry changed from "yes, category: application" to "yes, category: tuning". Firmware version FIR-v1748-B538662: "Name" entry changed from "Nominal Current" to "Motor Rated Current". Firmware version FIR-v1825-B577172: "Name" entry changed from "Motor Rated Current" to "Nominal Current". Firmware version FIR-v1825-B577172: "Name" entry changed from "Nominal Current" to "Motor Rated Current". Firmware version FIR-v1825-B577172: "Name" entry changed from "Motor Rated Current" to "Nominal Current". Firmware version FIR-v1825-B577172: "Name" entry changed from "Nominal Current" to "Motor Rated Current". Firmware version FIR-v1825-B577172: The number of entries was changed from 8 to 7. Firmware version FIR-v1926-B648637: "Name" entry changed from "Maximum Duration Of Peak Current" to "Maximum Duration Of Max Current".

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	06_h

Subindex	01_h
Name	Motor Rated Current
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000_h

Subindex	02_h

Name	Maximum Duration Of Max Current
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	$000003E8_h$

Subindex	03_h
Name	Threshold
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000_h

Subindex	04_h
Name	Calculated Value
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000_h

Subindex	05_h
Name	Limited Current
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000_h

Subindex	06_h
Name	Status
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000_h

Description

The subindices are divided into two groups: subindex 01_h and 02_h contain parameters for control, subindices 03_h to 06_h are status values. The functions are as follows:

- 01_h: The rated current specified in the motor data sheet is entered here in mA. This must be smaller than the current entered in 2031_h and 6073_h, otherwise monitoring is not activated. The specified value is interpreted as root mean square.
- 02_h: Specifies the maximum duration of the maximum current (6073_h) in ms.
- 03_h: Threshold, specifies the limit in A^2ms that determines whether the maximum current or rated current is switched to.
- 04_h: CalcValue, specifies the calculated value in A^2ms that is compared with the threshold for setting the current.
- 05_h: LimitedCurrent, contains the momentary current as root mean square set by I^2t.
- 06_h: Current status. If the sub-entry value is "0", I^2t is deactivated; if the value is "1", I^2t is activated.

203Dh Torque Window

Function

Specifies a symmetrical range relative to the target torque within which the target is considered having been met.

If the value is set to "FFFFFFFF"$_h$, monitoring is switched off, the "Target reached" bit in object 6041_h (statusword) is never set.

Object description

Index	$203D_h$
Object name	Torque Window
Object Code	VARIABLE
Data type	UNSIGNED16
Savable	yes, category: application
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	0000_h
Firmware version	FIR-v1540
Change history	Firmware version FIR-v1614: "Savable" entry changed from "no" to "yes, category: application".

203Eh Torque Window Time Out

Function

The current torque must be within the "Torque Window" ($203D_h$) for this time (in milliseconds) for the target torque to be considered having been met.

Object description

Index	$203E_h$
Object name	Torque Window Time Out
Object Code	VARIABLE
Data type	UNSIGNED16
Savable	yes, category: application
Access	read / write
PDO mapping	RX-PDO

Allowed values	
Preset value	0000_h
Firmware version	FIR-v1540
Change history	Firmware version FIR-v1614: "Savable" entry changed from "no" to "yes, category: application". Firmware version FIR-v1738-B501312: "Object Name" entry changed from "Torque Window Time" to "Torque Window Time Out".

203Fh Max Slippage Time Out

Function

Time in milliseconds until an excessively large slippage error in Profile Velocity mode results in an error message.

Object description

Index	$203F_h$
Object name	Max Slippage Time Out
Object Code	VARIABLE
Data type	UNSIGNED16
Savable	yes, category: application
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	0064_h
Firmware version	FIR-v1738-B501312
Change history	

Description

If the actual speed deviates so much from the set speed that the value (absolute value) of the object $60F8_h$ (Max Slippage) is exceeded, bit 13 in object 6041_h is set. The deviation must last longer than the time in object $203F_h$.

A reaction to the slippage error can be set in object 3700_h. If a reaction is defined, an error is also entered in object 1003_h.

2057h Clock Direction Multiplier

Function

The clock count value in Clock-direction mode is multiplied by this value before it is processed further.

Object description

Index	2057_h
Object name	Clock Direction Multiplier
Object Code	VARIABLE
Data type	UNSIGNED32

Savable	yes, category: application
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000080_h
Firmware version	FIR-v1426
Change history	Firmware version FIR-v2339-B1048823: "Data type" entry changed from "INTEGER32" to "UNSIGNED32".

2058h Clock Direction Divider

Function

The clock count value in Clock-direction mode is divided by this value before it is processed further.

Object description

Index	2058_h
Object name	Clock Direction Divider
Object Code	VARIABLE
Data type	UNSIGNED32
Savable	yes, category: application
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000001_h
Firmware version	FIR-v1426
Change history	Firmware version FIR-v2339-B1048823: "Data type" entry changed from "INTEGER32" to "UNSIGNED32".

205Ah Absolute Sensor Boot Value (in User Units)

Function

The initial encoder position when switching on the controller (in user-defined units) can be read from this object.

Object description

Index	$205A_h$
Object name	Absolute Sensor Boot Value (in User Units)
Object Code	VARIABLE
Data type	INTEGER32
Savable	no
Access	read only
PDO mapping	no
Allowed values	
Preset value	00000000_h

Firmware version	FIR-v1446
Change history	Firmware version FIR-v1512: "Access" table entry for subindex 00 changed from "read/write" to "read only". Firmware version FIR-v1738-B501312: "Object Name" entry changed from "Encoder Boot Value" to "Absolute Sensor Boot Value (in User Units)". Firmware version FIR-v1738-B501312: "Data type" entry changed from "UNSIGNED32" to "INTEGER32".

205Bh Clock Direction Or Clockwise/Counter Clockwise Mode

Function

This object can be used to switch the clock-direction mode (value = "0") to the right/left rotation mode (value = "1").

Object description

Index	$205B_h$
Object name	Clock Direction Or Clockwise/Counter Clockwise Mode
Object Code	VARIABLE
Data type	UNSIGNED32
Savable	yes, category: application
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000_h
Firmware version	FIR-v1504
Change history	

2084h Bootup Delay

Function

Defines the period between the time that supply voltage is applied to the controller and the functional readiness of the controller in milliseconds.

Object description

Index	2084_h
Object name	Bootup Delay
Object Code	VARIABLE
Data type	UNSIGNED32
Savable	yes, category: application
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000_h

Firmware version	FIR-v1426
Change history	

2101h Fieldbus Module Availability

Function

Shows the available fieldbuses.

Object description

Index	2101_h
Object name	Fieldbus Module Availability
Object Code	VARIABLE
Data type	UNSIGNED32
Savable	no
Access	read only
PDO mapping	no
Allowed values	
Preset value	■ PD1-C281S15-E-20-2: 00000008_h ■ PD1-C281S15-E-20-5: 00010002_h ■ PD1-C281S15-E-65-2: 00000008_h ■ PD1-C281S15-E-65-5: 00010002_h ■ PD1-C281S15-E-OF-2: 00000008_h ■ PD1-C281S15-E-OF-5: 00010002_h ■ PD1-C281L15-E-20-2: 00000008_h ■ PD1-C281L15-E-20-5: 00010002_h ■ PD1-C281L15-E-65-2: 00000008_h ■ PD1-C281L15-E-65-5: 00010002_h ■ PD1-C281L15-E-OF-2: 00000008_h ■ PD1-C281L15-E-OF-5: 00010002_h
Firmware version	FIR-v1426
Change history	Firmware version FIR-v1626: "Object Name" entry changed from "Fieldbus Module" to "Fieldbus Module Availability".

Description

Bits 0 to 15 represent the physical interface, bits 16 to 31 the used protocol (if necessary).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
													E-IP	MTCP	MRTU

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
									SPI	E-CAT	E-NET	CAN	RS232	RS485	USB

USB

Value = "1": The USB fieldbus is available.

RS-485

Value = "1": An RS-485 interface is available.

RS-232

Value = "1": An RS-232 interface is available.

CAN

Value = "1": The CANopen fieldbus is available.

E-NET

Value = "1": An Ethernet interface is available.

E-CAT

Value = "1": An EtherCAT interface is available.

SPI

Value = "1": An SPI interface is available.

MRTU

Value = "1": The used protocol is Modbus RTU.

MTCP

Value = "1": The used protocol is Modbus TCP

E-IP

Value = "1": The used protocol is EtherNet/IP™

2102h Fieldbus Module Control

Function

This object can be used to activate/deactivate certain fieldbuses (physical interfaces and protocols).

Object description

Index	2102_h
Object name	Fieldbus Module Control
Object Code	VARIABLE
Data type	UNSIGNED32
Savable	yes, category: communication
Access	read / write
PDO mapping	no
Allowed values	
Preset value	■ PD1-C281S15-E-20-2: 00000008_h ■ PD1-C281S15-E-20-5: 00010002_h ■ PD1-C281S15-E-65-2: 00000008_h ■ PD1-C281S15-E-65-5: 00010002_h ■ PD1-C281S15-E-OF-2: 00000008_h ■ PD1-C281S15-E-OF-5: 00010002_h ■ PD1-C281L15-E-20-2: 00000008_h ■ PD1-C281L15-E-20-5: 00010002_h ■ PD1-C281L15-E-65-2: 00000008_h ■ PD1-C281L15-E-65-5: 00010002_h ■ PD1-C281L15-E-OF-2: 00000008_h ■ PD1-C281L15-E-OF-5: 00010002_h
Firmware version	FIR-v1540

Change history	Firmware version FIR-v1626: "Savable" entry changed from "yes, category: application" to "yes, category: communication".

Description

Object 2103h:1h contains all physical interfaces/protocols that can be activated/deactivated. These can be switched in this object (2102h). The current status of the activated fieldbuses is in object 2103h:2h.

The following distribution of the bits applies here:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
													E-IP	MTCP	MRTU

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
									SPI	E-CAT	E-NET	CAN	RS232	RS485	USB

USB
USB interface

RS-485
RS-485 interface

RS-232
RS-232 interface

CAN
CANopen interface

E-NET
EtherNet interface

E-CAT
EtherCAT interface

SPI
SPI interface

MRTU
Modbus RTU protocol

MTCP
Modbus TCP protocol

E-IP
EtherNet/IP™ protocol

2103h Fieldbus Module Status

Function

Shows the active fieldbuses.

Object description

Index	2103h

Object name	Fieldbus Module Status
Object Code	ARRAY
Data type	UNSIGNED32
Savable	no
Access	read only
PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v1540
Change history	

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	02_h

Subindex	01_h
Name	Fieldbus Module Disable Mask
Data type	UNSIGNED32
Access	read only
PDO mapping	no
Allowed values	
Preset value	■ PD1-C281S15-E-20-2: 00000008_h ■ PD1-C281S15-E-20-5: 00010002_h ■ PD1-C281S15-E-65-2: 00000008_h ■ PD1-C281S15-E-65-5: 00010002_h ■ PD1-C281S15-E-OF-2: 00000008_h ■ PD1-C281S15-E-OF-5: 00010002_h ■ PD1-C281L15-E-20-2: 00000008_h ■ PD1-C281L15-E-20-5: 00010002_h ■ PD1-C281L15-E-65-2: 00000008_h ■ PD1-C281L15-E-65-5: 00010002_h ■ PD1-C281L15-E-OF-2: 00000008_h ■ PD1-C281L15-E-OF-5: 00010002_h

Subindex	02_h
Name	Fieldbus Module Enabled
Data type	UNSIGNED32
Access	read only
PDO mapping	no

Allowed values	
Preset value	00000000_h

Description

Subindex 1 (Fieldbus Module Disable Mask): This subindex contains all physical interfaces and protocols that can be activated or deactivated. The set bit means that this fieldbus can be deactivated.

Subindex 2 (Fieldbus Module Enabled): This subindex contains all currently activated physical interfaces and protocols. The set bit means that the fieldbus is active.

The following distribution of the bits applies for subindices 1 and 2:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
													E-IP	MTCP	MRTU

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
									SPI	E-CAT	E-NET	CAN	RS232	RS485	USB

USB
USB interface

RS-485
RS-485 interface

RS-232
RS-232 interface

CAN
CANopen interface

E-NET
EtherNet interface

E-CAT
EtherCAT interface

SPI
SPI interface

MRTU
Modbus RTU protocol

MTCP
Modbus TCP protocol

E-IP
EtherNet/IP™ protocol

2290h PDI Control

Function

With this object, you can activate the *Plug&Drive interface*. You can find additional information in document *Function description Plug&Drive interface*.

Object description

Index	2290_h
Object name	PDI Control
Object Code	VARIABLE
Data type	UNSIGNED8
Savable	yes, category: application
Access	read / write
PDO mapping	no
Allowed values	
Preset value	01_h
Firmware version	FIR-v1748-B531667
Change history	Firmware version FIR-v1748-B538662: "Access" table entry for subindex 00 changed from "read only" to "read/write".

Description

To activate the *Plug&Drive interface*, set bit 0 to "1".

2291h PDI Input

Function

If you use the *Plug&Drive interface*, you can use this object to select and start the operating mode and set the corresponding target values (target position, speed, etc.). You can find additional information in document *Function description Plug&Drive interface*.

Object description

Index	2291_h
Object name	PDI Input
Object Code	RECORD
Data type	PDI_INPUT
Savable	no
Access	read only
PDO mapping	RX-PDO
Allowed values	
Preset value	
Firmware version	FIR-v1748-B531667
Change history	Firmware version FIR-v2013-B726332: "Savable" entry changed from "yes, category: application" to "no". Firmware version FIR-v2315-B1040535: "Data type" entry changed from "INTEGER8" to "UNSIGNED8".

Value description

Subindex	00_h
Name	Number Of Entries

Data type	UNSIGNED8
Access	read only
PDO mapping	RX-PDO
Allowed values	
Preset value	04_h

Subindex	01_h
Name	PDI Set Value 1
Data type	INTEGER32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000000_h

Subindex	02_h
Name	PDI Set Value 2
Data type	INTEGER16
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	0000_h

Subindex	03_h
Name	PDI Set Value 3
Data type	INTEGER8
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00_h

Subindex	04_h
Name	PDI Command
Data type	UNSIGNED8
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00_h

2292h PDI Output

Function

If you use the *Plug&Drive interface*, you can, in this object, read the status and a return value that is dependent on the used operating mode. You can find additional information in document *Function description Plug&Drive interface*.

Object description

Index	2292_h
Object name	PDI Output
Object Code	RECORD
Data type	PDI_OUTPUT
Savable	no
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	
Firmware version	FIR-v1748-B531667
Change history	Firmware version FIR-v2315-B1040535: "Data type" entry changed from "INTEGER16" to "UNSIGNED16".

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	02_h

Subindex	01_h
Name	PDI Status
Data type	UNSIGNED16
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	0000_h

Subindex	02_h
Name	PDI Return Value
Data type	INTEGER32
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	00000000_h

2300h NanoJ Control

Function

Controls the execution of a *NanoJ program*.

Object description

Index	2300_h
Object name	NanoJ Control
Object Code	VARIABLE
Data type	UNSIGNED32
Savable	yes, category: application
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000000_h
Firmware version	FIR-v1426
Change history	Firmware version FIR-v1436: "Object Name" entry changed from "VMM Control" to "NanoJ Control".

Description

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
													AYield		ON

ON

Switches the *NanoJ program* on (value = "1") or off (value = "0").

With a rising edge in bit 0, the program is first reloaded and the variable range reset.

NOTICE

Startup of the *NanoJ program* can take up to 200 ms.

When switching on, a check is performed to determine whether a *NanoJ program* is present. If present, "1" is entered in 2300 and the *NanoJ program* is started.

AYield (AutoYield)

If this feature is activated (bit set to "1"), the *NanoJ program* is no longer stopped if it runs longer than it is allowed to. The *NanoJ program* is, thus, no longer real-time capable and no longer runs every 1 ms (see Available computing time).

NOTICE

Do not use the Debug output if *AutoYield* mode is activated.

2301h NanoJ Status

Function

Indicates the operating state of the user program.

Object description

Index	2301_h
Object name	NanoJ Status
Object Code	VARIABLE
Data type	UNSIGNED32
Savable	no
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	00000000_h
Firmware version	FIR-v1426
Change history	Firmware version FIR-v1436: "Object Name" entry changed from "VMM Status" to "NanoJ Status".

Description

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
													ERR	RES	RUN

RUN

Value = "0": Program is stopped, value = "1": NanoJ program is running.

RES

Reserved.

ERR

Program was ended with an error. Cause of the error can be read from object 2302_h.

2302h NanoJ Error Code

Function

Indicates which error occurred during the execution of the user program.

Object description

Index	2302_h
Object name	NanoJ Error Code
Object Code	VARIABLE
Data type	UNSIGNED32
Savable	no

Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	00000000_h
Firmware version	FIR-v1426
Change history	Firmware version FIR-v1436: "Object Name" entry changed from "VMM Error Code" to "NanoJ Error Code".

Description

Error codes during program execution:

Number	Description
0001_h	Firmware does not support the used function (e.g., `sin`, `cos`, etc.)
0002_h	Error in memory management
0005_h	Time Out: Code executed too long without `yield()` or `sleep()`
0100_h	Invalid NanoJ program file
0102_h	CRC error in the NanoJ program file

Error when accessing an object:

Number	Description
$1xxxxyy_h$	Invalid mapping in the NanoJ program file: The value in "xxxx" specifies the index, the value in "yy" specifies the subindex of the object that should – but cannot – be mapped.
2000000_h + number of surplus variables	Invalid mapping in the NanoJ program file: too many variables of type `input` were declared (see 2310h NanoJ Input Data Selection)
3000000_h + number of surplus variables	Invalid mapping in the NanoJ program file: too many variables of type `output` were declared (see 2320h NanoJ Output Data Selection)
4000000_h + number of surplus variables	Invalid mapping in the NanoJ program file: too many variables of type `inout` were declared (see 2330h NanoJ In/output Data Selection)
1000_h	Access of a nonexistent object in the object dictionary
1001_h	Write access of a write-protected entry in the OD
1002_h	An attempt was made to write a value that is too low or too high to an object.
1003_h	An attempt was made to read out an object that permits only write access.
$1FFF_h$	Unauthorized access of an object

2303h NanoJ Metadata

Function

This object is used to correctly recognize the NanoJ program by external software, e.g., *Plug & Drive Studio*.

Object description

Index	2303_h
Object name	NanoJ Metadata
Object Code	ARRAY

Data type	UNSIGNED32
Savable	no
Access	read only
PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v2339-B1048823
Change history	

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	06_h

Subindex	01_h
Name	Version
Data type	UNSIGNED32
Access	read only
PDO mapping	no
Allowed values	
Preset value	00000300_h

Subindex	02_h
Name	FLASH Start Address
Data type	UNSIGNED32
Access	read only
PDO mapping	no
Allowed values	
Preset value	00000000_h

Subindex	03_h
Name	FLASH End Address
Data type	UNSIGNED32
Access	read only
PDO mapping	no
Allowed values	
Preset value	00000000_h

Subindex	04_h
Name	RAM Start Address
Data type	UNSIGNED32
Access	read only
PDO mapping	no
Allowed values	
Preset value	00000000_h

Subindex	05_h
Name	RAM End Address
Data type	UNSIGNED32
Access	read only
PDO mapping	no
Allowed values	
Preset value	00000000_h

Subindex	06_h
Name	Header Size
Data type	UNSIGNED32
Access	read only
PDO mapping	no
Allowed values	
Preset value	00000000_h

230Eh Timer

Function

This object contains the operating time in milliseconds since the last time the controller was started.

NOTICE

This object is not stored; counting begins with "0" again after switching on or an overflow.

Object description

Index	$230E_h$
Object name	Timer
Object Code	ARRAY
Data type	UNSIGNED32
Savable	no
Access	read only
PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v2139-B1020888

Change history	

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	01_h

Subindex	01_h
Name	1ms Timer
Data type	UNSIGNED32
Access	read only
PDO mapping	no
Allowed values	
Preset value	00000000_h

230Fh Uptime Seconds

Function

This object contains the operating time in seconds since the last time the controller was started.

NOTICE

This object is not stored; counting begins with "0" again after switching on or an overflow.

Object description

Index	$230F_h$
Object name	Uptime Seconds
Object Code	VARIABLE
Data type	UNSIGNED32
Savable	no
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	00000000_h
Firmware version	FIR-v1436
Change history	

2310h NanoJ Input Data Selection

Function

Describes the object dictionary entries that are copied to the PDO mapping input of the NanoJ program.

Object description

Index	2310_h
Object name	NanoJ Input Data Selection
Object Code	ARRAY
Data type	UNSIGNED32
Savable	no
Access	read / write
PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v1650-B472161
Change history	Firmware version FIR-v1436: "Object Name" entry changed from "VMM Input Data Selection" to "NanoJ Input Data Selection". Firmware version FIR-v1650-B472161: "Savable" entry changed from "yes, category: application" to "no". Firmware version FIR-v1650-B472161: "Access" table entry for subindex 00 changed from "read/write" to "read only". Firmware version FIR-v1650-B472161: "Access" table entry for subindex 01 changed from "read/write" to "read only".

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	10_h

Subindex	01_h - 10_h
Name	Mapping #1 - #16
Data type	UNSIGNED32
Access	read only
PDO mapping	no
Allowed values	
Preset value	00000000_h

Description

Each subindex (1–16) describes a different mapped object.

A mapping entry consists of four bytes, which are structured according to the following graphic.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Index [16]															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SubIndex [8]								Length [8]							

Index [16]

This contains the index of the object to be mapped

Subindex [8]

This contains the subindex of the object to be mapped

Length [8]

This contains the length of the object to be mapped in units of bits.

2320h NanoJ Output Data Selection

Function

Describes the object dictionary entries that are copied into the output PDO mapping of the *NanoJ program* after it is executed.

Object description

Index	2320_h
Object name	NanoJ Output Data Selection
Object Code	ARRAY
Data type	UNSIGNED32
Savable	no
Access	read / write
PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v1650-B472161
Change history	Firmware version FIR-v1436: "Object Name" entry changed from "VMM Output Data Selection" to "NanoJ Output Data Selection". Firmware version FIR-v1650-B472161: "Savable" entry changed from "yes, category: application" to "no". Firmware version FIR-v1650-B472161: "Access" table entry for subindex 00 changed from "read/write" to "read only". Firmware version FIR-v1650-B472161: "Access" table entry for subindex 01 changed from "read/write" to "read only".

Value description

Subindex	00_h

Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	10_h

Subindex	01_h - 10_h
Name	Mapping #1 - #16
Data type	UNSIGNED32
Access	read only
PDO mapping	no
Allowed values	
Preset value	00000000_h

Description

Each subindex (1–16) describes a different mapped object.

A mapping entry consists of four bytes, which are structured according to the following graphic.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Index [16]															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SubIndex [8]								Length [8]							

Index [16]
: This contains the index of the object to be mapped

Subindex [8]
: This contains the subindex of the object to be mapped

Length [8]
: This contains the length of the object to be mapped in units of bits.

2330h NanoJ In/output Data Selection

Function

Describes the object dictionary entries that are first copied to the input PDO mapping of the NanoJ program and, after it is executed, are copied back to the output PDO mapping.

Object description

Index	2330_h
Object name	NanoJ In/output Data Selection
Object Code	ARRAY
Data type	UNSIGNED32
Savable	no
Access	read / write
PDO mapping	no

Allowed values	
Preset value	
Firmware version	FIR-v1650-B472161
Change history	Firmware version FIR-v1436: "Object Name" entry changed from "VMM In/output Data Selection" to "NanoJ In/output Data Selection". Firmware version FIR-v1650-B472161: "Savable" entry changed from "yes, category: application" to "no". Firmware version FIR-v1650-B472161: "Access" table entry for subindex 00 changed from "read/write" to "read only". Firmware version FIR-v1650-B472161: "Access" table entry for subindex 01 changed from "read/write" to "read only".

Value description

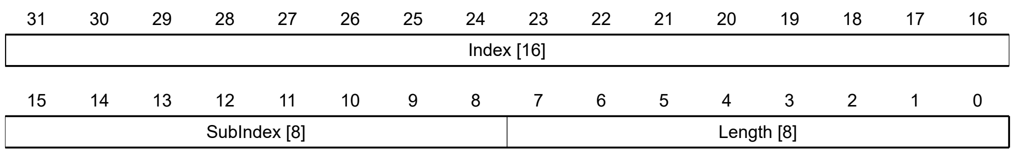
Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	10_h

Subindex	01_h - 10_h
Name	Mapping #1 - #16
Data type	UNSIGNED32
Access	read only
PDO mapping	no
Allowed values	
Preset value	00000000_h

Description

Each subindex (1–16) describes a different mapped object.

A mapping entry consists of four bytes, which are structured according to the following graphic.



Index [16]

This contains the index of the object to be mapped

Subindex [8]

This contains the subindex of the object to be mapped

Length [8]

This contains the length of the object to be mapped in units of bits.

2400h NanoJ Inputs

Function

Located here is an array with 32, 32-bit integer values that is not used within the firmware and serves only for communicating with the user program via the fieldbus.

Object description

Index	2400_h
Object name	NanoJ Inputs
Object Code	ARRAY
Data type	INTEGER32
Savable	no
Firmware version	FIR-v1426
Change history	The number of entries was changed from 2 to 33 Firmware version FIR-v1436: "Object Name" entry changed from "VMM Inputs" to "NanoJ Inputs". Firmware version FIR-v1436: "Name" entry changed from "VMM Input N#" to "NanoJ Input N#".

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	20_h

Subindex	01_h - 20_h
Name	NanoJ Input #1 - #32
Data type	INTEGER32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000000_h

Description

Here, it is possible to pass, e.g., preset values, to the *NanoJ program*.

2410h NanoJ Init Parameters

Function

This object functions identically to object 2400_h with the difference that this object can be stored.

Object description

Index	2410_h
Object name	NanoJ Init Parameters
Object Code	ARRAY
Data type	INTEGER32
Savable	yes, category: application
Access	read only
PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v1450
Change history	Firmware version FIR-v1450: "Data Type" entry changed from "INTEGER32" to "UNSIGNED8".

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	20_h

Subindex	01_h - 20_h
Name	NanoJ Init Parameter #1 - #32
Data type	INTEGER32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000000_h

2500h NanoJ Outputs

Function

Located here is an array with 32, 32-bit integer values that is not used within the firmware and serves only for communicating with the user program via the fieldbus.

Object description

Index	2500_h
Object name	NanoJ Outputs
Object Code	ARRAY
Data type	INTEGER32
Savable	no
Firmware version	FIR-v1426
Change history	Firmware version FIR-v1436: "Object Name" entry changed from "VMM Outputs" to "NanoJ Outputs". Firmware version FIR-v1436: "Name" entry changed from "VMM Output N#" to "NanoJ Output N#".

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	20_h

Subindex	01_h - 20_h
Name	NanoJ Output #1 - #32
Data type	INTEGER32
Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	00000000_h

Description

Here, the *NanoJ program* can store results which can then be read out via the fieldbus.

2600h NanoJ Debug Output

Function

This object contains debug output of a user program.

Object description

Index	2600_h
Object name	NanoJ Debug Output
Object Code	ARRAY
Data type	UNSIGNED8

Savable	no
Firmware version	FIR-v1426
Change history	Firmware version FIR-v1436: "Object Name" entry changed from "VMM Debug Output" to "NanoJ Debug Output".

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00_h

Subindex	01_h - 40_h
Name	Value #1 - #64
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	00_h

Description

Here, the NanoJ program stores the debug output that was called up with the `VmmDebugOutputString()` and `VmmDebugOutputInt()`.

2701h Customer Storage Area

Function

Data can be deposited and stored in this object.

Object description

Index	2701_h
Object name	Customer Storage Area
Object Code	ARRAY
Data type	UNSIGNED32
Savable	yes, category: customer
Access	read only
PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v1540

Change history	Firmware version FIR-v1540: "Data Type" entry changed from "UNSIGNED32" to "UNSIGNED8".

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	FE_h

Subindex	01_h - FE_h
Name	Storage #1 - #254
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000_h

2800h Bootloader And Reboot Settings

Function

Object description

Index	2800_h
Object name	Bootloader And Reboot Settings
Object Code	ARRAY
Data type	UNSIGNED32
Savable	yes, category: application
Access	read only
PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v1540
Change history	

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8

Access	read only
PDO mapping	no
Allowed values	
Preset value	03_h

Subindex	01_h
Name	Reboot Command
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000_h

Subindex	02_h
Name	Reboot Delay Time In Ms
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000064_h

Subindex	03_h
Name	Bootloader HW Config
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000_h

Description

The subindices have the following function:

- 01_h: If the value "746F6F62$_h$" is entered here, the firmware is rebooted.
- 02_h: Time in milliseconds: delays the reboot of the firmware by the respective time.

306C$_h$ Velocity Actual Internal Value

Function

Contains the current actual position in rpm.

Object description

Index	$306C_h$
Object name	Velocity Actual Internal Value
Object Code	VARIABLE

Data type	INTEGER32
Savable	no
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	00000000_h
Firmware version	FIR-v2412-B1057638
Change history	

3202h Motor Drive Submode Select

Function

Controls the controller mode, such as the changeover between *closed-loop* / *open-loop* and whether Velocity Mode is simulated via the S-controller or functions with a real V-controller in *closed loop*.

Object description

Index	3202_h
Object name	Motor Drive Submode Select
Object Code	VARIABLE
Data type	UNSIGNED32
Savable	yes, category: drive
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000001_h
Firmware version	FIR-v1426
Change history	Firmware version FIR-v1540: "Savable" entry changed from "yes category: application" to "yes, category: travel". Firmware version FIR-v1540: "Savable" entry changed from "yes category: travel" to "yes, category: movement".

Description

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								Slow	BLDC	Torque	AutoAl	CurRed	Brake	VoS	CL/OL

CL/OL

Changeover between *open-loop* and *closed-loop* (see chapter Control modes)

- Value = "0": *open-loop*
- Value = "1": *closed loop*

Toggling is not possible in the *Operation enabled* state.

VoS

Value = "1": Simulate V-controller with an S-ramp: simulate the speed modes through continuous position changes

Brake

Value = "1": Switch on automatic brake control.

CurRed (Current Reduction)

Current reduction in *open-loop*: Since version *FIR-v2213*, this remains activated provided that objects 2036_h and 2037_h are not both set to "0".

Torque

only active in operating modes Profile Torque and Cyclic Synchronous Torque

Value = "1": M-controller is active, otherwise a V-controller is superimposed: no V-controller is used in the torque modes for speed limiting, thus object 6080_h is ignored; 3210_h:3 and 3210_h:4 have no effect on the control.

BLDC

Value = "1": Motor type "BLDC" (brushless DC motor)

3203h Feedback Selection

Function

In this object, the sources of the presets are defined for the commutation and the velocity and position control.

A value change in the *Operation enabled* state shows no immediate effect. Value changes in objects are buffered and read out upon changing to the *Operation enabled* state.

Object description

Index	3203_h
Object name	Feedback Selection
Object Code	ARRAY
Data type	UNSIGNED8
Savable	yes, category: tuning
Access	read only
PDO mapping	RX-PDO
Allowed values	
Preset value	
Firmware version	FIR-v1748-B538662
Change history	

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	RX-PDO

Allowed values	
Preset value	02_h

Subindex	01_h
Name	1st Feedback Interface
Data type	UNSIGNED8
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00_h

Subindex	02_h
Name	2nd Feedback Interface
Data type	UNSIGNED8
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	07_h

Description

The subindices have the following function:

- 00_h: Value="1" to "n", where "n" is the number of existing feedbacks.
- n_h:
  Subindex n contains a bit mask for the respective feedback n. The bits have the following meaning here:
- Bit 0: If the bit is set to "1", this sensor is used for position feedback.
- Bit 1: If the bit is set to "1", this sensor is used for velocity feedback.
- Bit 2: If the bit is set to "1", this sensor is used for commutation feedback in Closed-Loop.

Subindex 01_h always corresponds to the first (and always existing) *sensorless* feedback.

Which sensor the controller takes into account for the individual controllers (commutation, velocity, position) is implicitly specified by the order of the sensors.

The search always begins with sensor 2 and continues in ascending order until all existing sensors have been queried. If a sensor is found whose feedback is set, it is assigned to the corresponding controller and the search ended.

NOTICE

If bit 0 in 3202_h is set to "0", *closed loop* is deactivated; bit 2 (commutation) then has no meaning. Bit 1 for the velocity and bit 0 for the position in the respective subindices are still used for the display of the actual position and speed values.

3204h Feedback Mapping

Function

This object contains information on the existing feedbacks.

Object description

Index	3204_h
Object name	Feedback Mapping
Object Code	ARRAY
Data type	UNSIGNED16
Savable	no
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	
Firmware version	FIR-v1748-B538662
Change history	

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	02_h

Subindex	01_h
Name	Index Of 1st Feedback Interface
Data type	UNSIGNED16
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	3380_h

Subindex	02_h
Name	Index Of 2nd Feedback Interface
Data type	UNSIGNED16
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	$33A0_h$

Description

The subindices have the following function:

- 00_h: Value="1" to "n", where "n" is the number of existing feedbacks.
- n_h:

Subindex n refers to the index of the respective object for the configuration of the corresponding feedback.
Subindex 01_h always corresponds to the first (and always existing) *sensorless* feedback.

3205_h Feedback Use

Function

This object shows, which sensor is used for which control loop.

Object description

Index	3205_h
Object name	Feedback Use
Object Code	ARRAY
Data type	UNSIGNED16
Savable	no
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	
Firmware version	FIR-v2412-B1057638
Change history	

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	04_h

Subindex	01_h
Name	Sensor For Position Control Loop
Data type	UNSIGNED16
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	0000_h

Subindex	02_h
Name	Sensor For Velocity Control Loop
Data type	UNSIGNED16
Access	read only
PDO mapping	TX-PDO

Allowed values	
Preset value	0000_h

Subindex	03_h
Name	Sensor For Current Control Loop
Data type	UNSIGNED16
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	0000_h

Subindex	04_h
Name	Sensor For Pulse Generator
Data type	UNSIGNED16
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	0000_h

320Dh Torque Of Inertia Factor

Function

This factor is used for calculating the acceleration feed forward (see 321D). Default is 0 (feed forward inactive).

Acceleration feed forward applies during deceleration as well.

Object description

Index	$320D_h$
Object name	Torque Of Inertia Factor
Object Code	ARRAY
Data type	UNSIGNED32
Savable	yes, category: drive
Access	read only
PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v1825-B577172
Change history	

Value description

Subindex	00_h
Name	Number Of Entries

Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	02_h

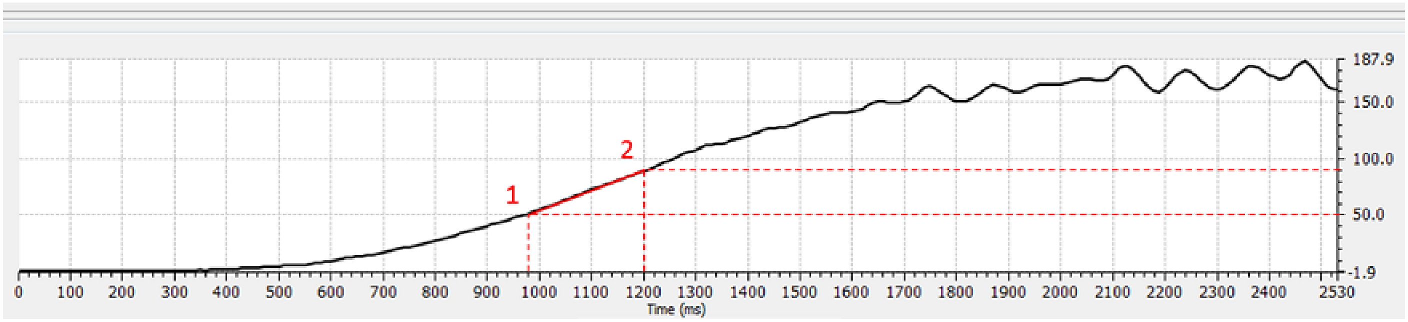
Subindex	01_h
Name	Current
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000_h

Subindex	02_h
Name	Acceleration
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000_h

Description

The value is dependent on the inertia of the load. To determine the factor:

1. Activate *closed loop* and select the *profile torque* mode.
2. Set a target for the torque and enter the corresponding current value (mA) in $320D_h$:01_h.
3. Record (e. g., in *Plug & Drive Studio*) the current speed (object $606C_h$). Calculate the acceleration in the set user-defined units for the speed range, where this is constant. Enter the value in $320D_h$:02_h.
   Using the speed curve in the following figure as an example:
   (90-50)/(1200-980)=182 rpm/s.



3212h Motor Drive Flags

Function

This object is used to specify whether or not auto setup is to adapt the controller parameters. The direction of the rotating field can also be changed.

NOTICE

Changes in subindex 02_h do not take effect until after the controller is restarted. Afterwards, Auto setup must again be performed.

Object description

Index	3212_h
Object name	Motor Drive Flags
Object Code	ARRAY
Data type	INTEGER8
Savable	yes, category: application
Access	read only
PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v1450
Change history	Firmware version FIR-v1512: The number of entries was changed from 2 to 3. Firmware version FIR-v1738-B501312: "Name" entry changed from "Enable Legacy Power Mode" to "Reserved".

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	03_h

Subindex	01_h
Name	Reserved
Data type	INTEGER8
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00_h

Subindex	02_h
Name	Override Field Inversion
Data type	INTEGER8
Access	read / write
PDO mapping	no

Allowed values	
Preset value	00_h

Subindex	03_h
Name	Auto-setup With Current Controller Parameters From The OD
Data type	INTEGER8
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00_h

Description

Valid values for subindex 02_h:

- Value = "0": Use default values of the firmware
- Value = "1": Force non-inversion of the rotating field (mathematically positive)
- Value = "-1": Force inversion of the rotating field (mathematically negative)

Valid values for subindex 03_h:

- Value = "0": Auto setup detects the motor type (stepper motor or BLDC motor) and uses the corresponding pre-configured parameter set.
- Value = "1": Perform auto setup with the control parameters that were entered in object 3210_h or $321A_h$ to $321E_h$ before the auto setup. The control parameters are not changed.

321Ah Current Controller Parameters

Function

Contains the parameters for the current controller (commutation). As a rule, the values for Iq (subindex 01_h/02_h) and Id (subindex 03_h/04_h) should be the same. See chapter Controller structure.

Object description

Index	$321A_h$
Object name	Current Controller Parameters
Object Code	ARRAY
Data type	UNSIGNED32
Savable	yes, category: application
Access	read only
PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v2213-B1028181
Change history	

Value description

Subindex	00_h
Name	Number Of Entries

Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	04_h

Subindex	01_h
Name	Proportional Gain Kp For Iq [mV/A]
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	■ PD1-C281S15-E-20-2: $00000F76_h$ ■ PD1-C281S15-E-20-5: $00000F76_h$ ■ PD1-C281S15-E-65-2: $00000F76_h$ ■ PD1-C281S15-E-65-5: $00000F76_h$ ■ PD1-C281S15-E-OF-2: $00000F76_h$ ■ PD1-C281S15-E-OF-5: $00000F76_h$ ■ PD1-C281L15-E-20-2: $00001DB0_h$ ■ PD1-C281L15-E-20-5: $00001DB0_h$ ■ PD1-C281L15-E-65-2: $00001DB0_h$ ■ PD1-C281L15-E-65-5: $00001DB0_h$ ■ PD1-C281L15-E-OF-2: $00001DB0_h$ ■ PD1-C281L15-E-OF-5: $00001DB0_h$

Subindex	02_h
Name	Integrator Time Ti For Iq [µs]
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	■ PD1-C281S15-E-20-2: $0000032B_h$ ■ PD1-C281S15-E-20-5: $0000032B_h$ ■ PD1-C281S15-E-65-2: $0000032B_h$ ■ PD1-C281S15-E-65-5: $0000032B_h$ ■ PD1-C281S15-E-OF-2: $0000032B_h$ ■ PD1-C281S15-E-OF-5: $0000032B_h$ ■ PD1-C281L15-E-20-2: $000003E8_h$ ■ PD1-C281L15-E-20-5: $000003E8_h$ ■ PD1-C281L15-E-65-2: $000003E8_h$ ■ PD1-C281L15-E-65-5: $000003E8_h$ ■ PD1-C281L15-E-OF-2: $000003E8_h$ ■ PD1-C281L15-E-OF-5: $000003E8_h$

Subindex	03_h
Name	Proportional Gain Kp For Id [mV/A]

Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	■ PD1-C281S15-E-20-2: $00000F76_h$ ■ PD1-C281S15-E-20-5: $00000F76_h$ ■ PD1-C281S15-E-65-2: $00000F76_h$ ■ PD1-C281S15-E-65-5: $00000F76_h$ ■ PD1-C281S15-E-OF-2: $00000F76_h$ ■ PD1-C281S15-E-OF-5: $00000F76_h$ ■ PD1-C281L15-E-20-2: $00001DB0_h$ ■ PD1-C281L15-E-20-5: $00001DB0_h$ ■ PD1-C281L15-E-65-2: $00001DB0_h$ ■ PD1-C281L15-E-65-5: $00001DB0_h$ ■ PD1-C281L15-E-OF-2: $00001DB0_h$ ■ PD1-C281L15-E-OF-5: $00001DB0_h$

Subindex	04_h
Name	Integrator Time Ti For Id [µs]
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	■ PD1-C281S15-E-20-2: $0000032B_h$ ■ PD1-C281S15-E-20-5: $0000032B_h$ ■ PD1-C281S15-E-65-2: $0000032B_h$ ■ PD1-C281S15-E-65-5: $0000032B_h$ ■ PD1-C281S15-E-OF-2: $0000032B_h$ ■ PD1-C281S15-E-OF-5: $0000032B_h$ ■ PD1-C281L15-E-20-2: $000003E8_h$ ■ PD1-C281L15-E-20-5: $000003E8_h$ ■ PD1-C281L15-E-65-2: $000003E8_h$ ■ PD1-C281L15-E-65-5: $000003E8_h$ ■ PD1-C281L15-E-OF-2: $000003E8_h$ ■ PD1-C281L15-E-OF-5: $000003E8_h$

321Bh Velocity Controller Parameters

Function

Contains the parameters for the velocity controller. See chapter Controller structure.

Object description

Index	$321B_h$
Object name	Velocity Controller Parameters
Object Code	ARRAY
Data type	UNSIGNED32

Savable	yes, category: application
Access	read only
PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v2213-B1028181
Change history	

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	02_h

Subindex	01_h
Name	Proportional Gain Kp [mA/Hz]
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000032_h

Subindex	02_h
Name	Integrator Time Ti [µs]
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	■ PD1-C281S15-E-20-2: $000061A8_h$ ■ PD1-C281S15-E-20-5: $000061A8_h$ ■ PD1-C281S15-E-65-2: $000061A8_h$ ■ PD1-C281S15-E-65-5: $000061A8_h$ ■ PD1-C281S15-E-OF-2: $000061A8_h$ ■ PD1-C281S15-E-OF-5: $000061A8_h$ ■ PD1-C281L15-E-20-2: $00004B00_h$ ■ PD1-C281L15-E-20-5: $00004B00_h$ ■ PD1-C281L15-E-65-2: $00004B00_h$ ■ PD1-C281L15-E-65-5: $00004B00_h$ ■ PD1-C281L15-E-OF-2: $00004B00_h$ ■ PD1-C281L15-E-OF-5: $00004B00_h$

321Ch Position Controller Parameters

Function

Contains the parameters for the position controller. See chapter Controller structure.

Object description

Index	$321C_h$
Object name	Position Controller Parameters
Object Code	ARRAY
Data type	UNSIGNED32
Savable	yes, category: application
Access	read only
PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v2213-B1028181
Change history	

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	02_h

Subindex	01_h
Name	Proportional Gain Kp [Hz]
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	■ PD1-C281S15-E-20-2: 00002710_h ■ PD1-C281S15-E-20-5: 00002710_h ■ PD1-C281S15-E-65-2: 00002710_h ■ PD1-C281S15-E-65-5: 00002710_h ■ PD1-C281S15-E-OF-2: 00002710_h ■ PD1-C281S15-E-OF-5: 00002710_h ■ PD1-C281L15-E-20-2: $0000C350_h$ ■ PD1-C281L15-E-20-5: $0000C350_h$ ■ PD1-C281L15-E-65-2: $0000C350_h$ ■ PD1-C281L15-E-65-5: $0000C350_h$ ■ PD1-C281L15-E-OF-2: $0000C350_h$

- PD1-C281L15-E-OF-5: $0000C350_h$

Subindex	02_h
Name	Integrator Time Ti [µs]
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000_h

321Dh Feedforward

Function

Contains the parameters for the feed forward. See chapter Controller structure.

Object description

Index	$321D_h$
Object name	Feedforward
Object Code	ARRAY
Data type	UNSIGNED16
Savable	yes, category: application
Access	read only
PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v2213-B1028181
Change history	Firmware version FIR-v2315-B1040535: "Data type" entry changed from "UNSIGNED32" to "UNSIGNED16". Firmware version FIR-v2315-B1040535: "Data type" entry changed from "UNSIGNED32" to "UNSIGNED16". Firmware version FIR-v2315-B1040535: "Data type" entry changed from "UNSIGNED32" to "UNSIGNED16". Firmware version FIR-v2315-B1040535: "Data type" entry changed from "UNSIGNED32" to "UNSIGNED16". Firmware version FIR-v2412-B1057638: entry "Object Name" changed from "Pre-control" to "Feedforward". Firmware version FIR-v2412-B1057638: entry "Name" changed from "Voltage Pre-control For Dq-decoupling [‰]" to "Voltage Feedforward For Dq-decoupling [‰]". Firmware version FIR-v2412-B1057638: entry "Name" changed from "Acceleration Pre-control [‰]" to "Acceleration Feedforward [‰]". Firmware version FIR-v2412-B1057638: entry "Name" changed from "Velocity Pre-control [‰]" to "Velocity Feedforward [‰]".

	Firmware version FIR-v2412-B1057638: entry "Name" changed from "Ohmic Based Voltage Pre-control [‰]" to "Ohmic Based Voltage Feedforward [‰]".

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	04_h

Subindex	01_h
Name	Voltage Feedforward For Dq-decoupling [‰]
Data type	UNSIGNED16
Access	read / write
PDO mapping	no
Allowed values	
Preset value	$03E8_h$

Subindex	02_h
Name	Acceleration Feedforward [‰]
Data type	UNSIGNED16
Access	read / write
PDO mapping	no
Allowed values	
Preset value	$03E8_h$

Subindex	03_h
Name	Velocity Feedforward [‰]
Data type	UNSIGNED16
Access	read / write
PDO mapping	no
Allowed values	
Preset value	$03E8_h$

Subindex	04_h
Name	Ohmic Based Voltage Feedforward [‰]
Data type	UNSIGNED16
Access	read / write
PDO mapping	no
Allowed values	

Preset value	0000_h

321Eh Voltage Limit

Function

Maximum permissible PWM voltage (duty cycle). Values ≤ 1000 are interpreted as per mil values (of the available voltage). Values > 1000 as millivolt. See also chapter Controller structure.

Object description

Index	$321E_h$
Object name	Voltage Limit
Object Code	VARIABLE
Data type	UNSIGNED32
Savable	yes, category: application
Access	read / write
PDO mapping	no
Allowed values	
Preset value	$000186A0_h$
Firmware version	FIR-v2213-B1028181
Change history	

Description

Also dependent on this value is whether the *overmodulation* of the voltage vector is used. If *overmodulation* is used, a higher torque can be achieved. The resulting voltage is no longer sinusoidal, which can result in harmonics and higher losses.

Value in mV	Overmodulation
1001…U_{o_low}	None; the voltage vector describes a circle.
U_{o_low}…U_{o_high}	The voltage vector describes a circle that is increasingly flattened on four/six sides in proportion to the set value.
≥U_{o_high}	Full; the voltage vector describes a square or a hexagon.

U_{o_low}

The lowest voltage above which overmodulation occurs. Is calculated as follows:

Operating voltage*0.9425

U_{o_high}

The maximum overmodulation occurs above this voltage. Is calculated as follows:

With two-phase stepper motors: operating voltage*1.063

With three-phase BLDC motors: operating voltage*0.99

3220h Analog Input Digits

Function

Displays the instantaneous values of the analog inputs in *ADC digits*.

Object description

Index	3220_h
Object name	Analog Input Digits
Object Code	ARRAY
Data type	INTEGER16
Savable	no
Access	read only
PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v1426
Change history	

Value description

Subindex	00_h
Name	Number Of Analog Input Digits
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	01_h

Subindex	01_h
Name	Analog Input #1 Digit
Data type	INTEGER16
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	0000_h

Description

Formulas for converting from [digits] to the respective unit:

- Current input (if configurable): x digits * 20 mA / 1023 digits

323Ah User Pin Settings

Function

With this object, you can configure the digital inputs/outputs as described in chapter Digital inputs and outputs.

Object description

Index	$323A_h$
Object name	User Pin Settings
Object Code	ARRAY
Data type	UNSIGNED8
Savable	yes, category: application
Access	read only
PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v2339-B1048823
Change history	

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	02_h

Subindex	01_h
Name	Voltage Level Select
Data type	UNSIGNED8
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00_h

Subindex	02_h
Name	Pull-Up Enable
Data type	UNSIGNED8
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00_h

Description

- Subindex 01_h: Here, you define the level for the inputs/outputs:
  - Value "0": 5 V
  - Value "1": 24 V (inputs) or +UB (outputs)

NOTICE

Use for the inputs a voltage that is smaller than the voltage +UB .

- Subindex 02_h: Here, you define the type of digital inputs:
  - Value "0" (Pull-Down): High level when 5/24 V at Pin.
  - Value "1" (Pull-Up): High level without external voltage at Pin.

3241h Digital Input Position Capture

Function

With this object, the encoder position can be noted automatically if a level change occurs at digital input .

Object description

Index	3241_h
Object name	Digital Input Position Capture
Object Code	ARRAY
Data type	UNSIGNED32
Savable	yes, category: application
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	
Firmware version	FIR-v1446
Change history	Firmware version FIR-v1446: "Data type" entry changed from "UNSIGNED32" to "UNSIGNED8". Firmware version FIR-v1738-B501312: "Name" entry changed from "Encoder Raw Value" to "Sensor Raw Value". Firmware version FIR-v1748-B531667: "PDO mapping" table entry for subindex 00 changed from "no" to "TX-PDO". Firmware version FIR-v1748-B531667: "PDO mapping" table entry for subindex 01 changed from "RX-PDO" to "TX-PDO". Firmware version FIR-v1748-B531667: "PDO mapping" table entry for subindex 02 changed from "RX-PDO" to "TX-PDO". Firmware version FIR-v1748-B531667: "PDO mapping" table entry for subindex 03 changed from "RX-PDO" to "TX-PDO". Firmware version FIR-v1748-B531667: "PDO mapping" table entry for subindex 04 changed from "RX-PDO" to "TX-PDO".

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	TX-PDO

Allowed values	
Preset value	04_h

Subindex	01_h - 04_h
Name	Control For Capture Of Input #1 - #4
Data type	UNSIGNED32
Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	00000000_h

Description

- Subindex 01_h: This is used to select the type of level change:
  - Deactivate function: Value "0"
  - With rising edge: Value "1"
  - With falling edge: Value "2"
  - Both edges: Value "3"
- Subindex 02_h: Specifies the number of the noted level changes since the time the function was started; is reset to 0 if subindex 01_h is set to 1,2 or 3
- Subindex 03_h: Encoder position of the level change (in absolute user units from 6064_h)
- Subindex 04_h: Encoder position of the level change

3242h Digital Input Routing

Function

This object determines the source of the input routing that ends in $60FD_h$.

Object description

Index	3242_h
Object name	Digital Input Routing
Object Code	ARRAY
Data type	UNSIGNED8
Savable	yes, category: application
Access	read only
PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v1504
Change history	

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8

Access	read only
PDO mapping	no
Allowed values	
Preset value	20_h

Subindex	01_h
Name	Input Source For Bit #0 In 60FDh - Negative Limit Switch
Data type	UNSIGNED8
Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	00_h

Subindex	02_h
Name	Input Source For Bit #1 In 60FDh - Positive Limit Switch
Data type	UNSIGNED8
Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	00_h

Subindex	03_h
Name	Input Source For Bit #2 In 60FDh - Home Switch
Data type	UNSIGNED8
Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	00_h

Subindex	04_h
Name	Input Source For Bit #3 In 60FDh - Interlock
Data type	UNSIGNED8
Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	00_h

Subindex	05_h
Name	Input Source For Bit #4 In 60FDh
Data type	UNSIGNED8
Access	read / write
PDO mapping	TX-PDO
Allowed values	

Preset value	00_h

Subindex	06_h
Name	Input Source For Bit #5 In 60FDh
Data type	UNSIGNED8
Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	00_h

Subindex	07_h
Name	Input Source For Bit #6 In 60FDh
Data type	UNSIGNED8
Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	00_h

Subindex	08_h
Name	Input Source For Bit #7 In 60FDh
Data type	UNSIGNED8
Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	00_h

Subindex	09_h
Name	Input Source For Bit #8 In 60FDh
Data type	UNSIGNED8
Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	00_h

Subindex	$0A_h$
Name	Input Source For Bit #9 In 60FDh
Data type	UNSIGNED8
Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	00_h

Subindex	$0B_h$

Name	Input Source For Bit #10 In 60FDh
Data type	UNSIGNED8
Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	00_h

Subindex	$0C_h$
Name	Input Source For Bit #11 In 60FDh
Data type	UNSIGNED8
Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	00_h

Subindex	$0D_h$
Name	Input Source For Bit #12 In 60FDh
Data type	UNSIGNED8
Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	00_h

Subindex	$0E_h$
Name	Input Source For Bit #13 In 60FDh
Data type	UNSIGNED8
Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	00_h

Subindex	$0F_h$
Name	Input Source For Bit #14 In 60FDh
Data type	UNSIGNED8
Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	00_h

Subindex	10_h
Name	Input Source For Bit #15 In 60FDh
Data type	UNSIGNED8
Access	read / write

PDO mapping	TX-PDO
Allowed values	
Preset value	00_h

Subindex	11_h
Name	Input Source For Bit #16 In 60FDh
Data type	UNSIGNED8
Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	01_h

Subindex	12_h
Name	Input Source For Bit #17 In 60FDh
Data type	UNSIGNED8
Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	02_h

Subindex	13_h
Name	Input Source For Bit #18 In 60FDh
Data type	UNSIGNED8
Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	03_h

Subindex	14_h
Name	Input Source For Bit #19 In 60FDh
Data type	UNSIGNED8
Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	04_h

Subindex	15_h
Name	Input Source For Bit #20 In 60FDh
Data type	UNSIGNED8
Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	05_h

Subindex	16_h
Name	Input Source For Bit #21 In 60FDh
Data type	UNSIGNED8
Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	06_h

Subindex	17_h
Name	Input Source For Bit #22 In 60FDh
Data type	UNSIGNED8
Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	$5A_h$

Subindex	18_h
Name	Input Source For Bit #23 In 60FDh
Data type	UNSIGNED8
Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	$5B_h$

Subindex	19_h
Name	Input Source For Bit #24 In 60FDh
Data type	UNSIGNED8
Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	07_h

Subindex	$1A_h$
Name	Input Source For Bit #25 In 60FDh
Data type	UNSIGNED8
Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	08_h

Subindex	$1B_h$
Name	Input Source For Bit #26 In 60FDh
Data type	UNSIGNED8

Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	09_h

Subindex	$1C_h$
Name	Input Source For Bit #27 In 60FDh
Data type	UNSIGNED8
Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	$0A_h$

Subindex	$1D_h$
Name	Input Source For Bit #28 In 60FDh
Data type	UNSIGNED8
Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	$0B_h$

Subindex	$1E_h$
Name	Input Source For Bit #29 In 60FDh
Data type	UNSIGNED8
Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	$0C_h$

Subindex	$1F_h$
Name	Input Source For Bit #30 In 60FDh
Data type	UNSIGNED8
Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	$0D_h$

Subindex	20_h
Name	Input Source For Bit #31 In 60FDh
Data type	UNSIGNED8
Access	read / write
PDO mapping	TX-PDO
Allowed values	

Preset value	$0E_h$

3243h Home Switch Position Capture

Function

With this object, the current position can be noted automatically if a level change occurs at the digital input that is used for the home switch.

NOTICE

Do not use this function in combination with a homing operation. The homing operation cannot otherwise be successfully completed.

Object description

Index	3243_h
Object name	Home Switch Position Capture
Object Code	ARRAY
Data type	UNSIGNED32
Savable	yes, category: application
Access	read only
PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v1738-B501312
Change history	Firmware version FIR-v2315-B1040535: "Object Name" entry changed from "Digital Input Homing Capture" to "Home Switch Position Capture".

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	04_h

Subindex	01_h
Name	Control
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	

Preset value	00000000$_h$

Subindex	02$_h$
Name	Capture Count
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000000$_h$

Subindex	03$_h$
Name	Value
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000000$_h$

Subindex	04$_h$
Name	Sensor Raw Value
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000000$_h$

Description

- Subindex 01$_h$: This is used to select the type of level change:
  - Deactivate function: Value "0"
  - With rising edge: Value "1"
  - With falling edge: Value "2"
  - Both edges: Value "3"
- Subindex 02$_h$: Specifies the number of the noted level changes since the time the function was started; is reset to 0 if subindex 01$_h$ is set to 1,2 or 3
- Subindex 03$_h$: Encoder position of the level change (in absolute user units from 6064$_h$)
- Subindex 04$_h$: Encoder position of the level change

324Ah Inputs

Function

You read out the current status of the inputs from this object.

Object description

Index	324A$_h$

Object name	Inputs
Object Code	ARRAY
Data type	UNSIGNED16
Savable	no
Access	read only
PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v2339-B1048823
Change history	

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	02_h

Subindex	01_h
Name	User Inputs
Data type	UNSIGNED16
Access	read only
PDO mapping	no
Allowed values	
Preset value	0000_h

Subindex	02_h
Name	Encoder Inputs
Data type	UNSIGNED16
Access	read only
PDO mapping	no
Allowed values	
Preset value	0000_h

Description

- Subindex 01_h: Shows the status of the following inputs:
  - Bits 0 to 5: Digital inputs 1 to 6 (exact number dependent on product)
  - Bits 6 and 7: Analog inputs 1 and 2 as digital input (exact number dependent on product)
- Subindex 02_h: Shows the status of the following sensor inputs:
  - Bits 0 to 2: Hall sensors 1 to 3 (if present)
  - Bits 3 to 5: Channels A, B, index of the first incremental encoder (if present)

- Bits 6 to 8: Channels A, B index of the second incremental encoder (if present)

3250h Digital Outputs Control

Function

NOTICE

This object has no function with this product. See also chapter " Digital inputs and outputs".

Object description

Index	3250_h
Object name	Digital Outputs Control
Object Code	ARRAY
Data type	UNSIGNED32
Savable	yes, category: application
Access	read only
PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v1426
Change history	Firmware version FIR-v1426: Subindex 01_h: "Name" entry changed from "Special Function Disable" to "Special Function Enable" Firmware version FIR-v1446: "Name" entry changed from "Special Function Enable" to "No Function". Firmware version FIR-v1512: The number of entries was changed from 6 to 9. Firmware version FIR-v2039: Subindex 09 added

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	01_h

Subindex	01_h
Name	Enable Mask [Bit0=StatusLed, Bit1=ErrorLed]
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO

Allowed values	
Preset value	FFFFFFFF$_h$

3252h Digital Output Routing

Function

This object assigns a signal source to an output; this signal source can be controlled with 60FE$_h$. You can find details in chapter *Output Routing*.

Object description

Index	3252$_h$
Object name	Digital Output Routing
Object Code	ARRAY
Data type	UNSIGNED16
Savable	yes, category: application
Access	read only
PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v1540
Change history	

Value description

Subindex	00$_h$
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	03$_h$

Subindex	01$_h$
Name	Control Bit Of 60FEh:1h And Source For Brake Output
Data type	UNSIGNED16
Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	1080$_h$

Subindex	02$_h$
Name	Control Bit Of 60FEh:1h And Source For Output #1
Data type	UNSIGNED16
Access	read / write

PDO mapping	TX-PDO
Allowed values	
Preset value	0010_h

Subindex	03_h
Name	Control Bit Of 60FEh:1h And Source For Output #2
Data type	UNSIGNED16
Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	0011_h

325Ah Outputs

Function

Use this object to control the digital outputs (alternative to 60FEh Digital Outputs).

Object description

Index	$325A_h$
Object name	Outputs
Object Code	ARRAY
Data type	UNSIGNED16
Savable	yes, category: application
Access	read only
PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v2339-B1048823
Change history	Firmware version FIR-v2412-B1057638: entry "Savable" changed from "yes, category: communication" to "yes, category: application".

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	01_h

Subindex	01_h
Name	User Outputs

Data type	UNSIGNED16
Access	read / write
PDO mapping	no
Allowed values	
Preset value	0000_h

Description

- Subindex 01_h:
  - Bits 0 to 6: Digital outputs 1 to 7 (exact number dependent on product)
  - Bit 7: Brake output (if present)

3273h Generic SPI Hardware Configuration

Function

See chapter Generic SPI.

Object description

Index	3273_h
Object name	Generic SPI Hardware Configuration
Object Code	ARRAY
Data type	UNSIGNED32
Savable	yes, category: application
Access	read only
PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v2213-B1029645
Change history	

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	01_h

Subindex	01_h
Name	Hardware Feature Control
Data type	UNSIGNED32
Access	read / write
PDO mapping	no

Allowed values	
Preset value	00000000$_h$

3274h Generic SPI Mosi Data

Function

See chapter Generic SPI.

Object description

Index	3274$_h$
Object name	Generic SPI Mosi Data
Object Code	ARRAY
Data type	UNSIGNED8
Savable	yes, category: application
Access	read / write
PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v2213-B1029645
Change history	

Value description

Subindex	00$_h$
Name	Length Of SPI Message To Be Sent
Data type	UNSIGNED8
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00$_h$

Subindex	01$_h$ - 1F$_h$
Name	Generic SPI Mosi Data Byte #1 - #31
Data type	UNSIGNED8
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00$_h$

3275h Generic SPI Miso Data

Function

See chapter Generic SPI.

Object description

Index	3275_h
Object name	Generic SPI Miso Data
Object Code	ARRAY
Data type	UNSIGNED8
Savable	no
Access	read only
PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v2213-B1029645
Change history	

Value description

Subindex	00_h
Name	Length Of Received SPI Message
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	00_h

Subindex	01_h - $1F_h$
Name	Generic SPI Miso Data Byte #1 - #31
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	00_h

3320h Analog Input Values

Function

This object displays the instantaneous values of the analog inputs in user-defined units.

Object description

Index	3320_h
Object name	Analog Input Values
Object Code	ARRAY
Data type	INTEGER32
Savable	no
Access	read only

PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v1426
Change history	

Value description

Subindex	00_h
Name	Number Of Analog Input Values
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	01_h

Subindex	01_h
Name	Analog Input #1 Value
Data type	INTEGER32
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	00000000_h

Description

The user-defined units are made up of offset (3321_h) and scaling value (3322_h/ 3323_h). If both are still set to the default values, the value in 3320_h is specified in the *ADC Digits* unit.

Formula for converting from digits to the respective unit:

- Current input (if configurable): x digits * 20 mA / 1023 digits

The following applies for the sub-entries:

- Subindex 00_h: Number of analog inputs
- Subindex 01_h: Analog value 1
- Subindex 02_h: Analog value 2 (if present)

3321h Analog Input Offsets

Function

Offset that is added to the read analog value (3220_h) before scaling (multiplier from object 3322 and divisor from object 3323_h).

Object description

Index	3321_h
Object name	Analog Input Offsets
Object Code	ARRAY

Data type	INTEGER16
Savable	yes, category: application
Access	read only
PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v2139-B1022383
Change history	

Value description

Subindex	00_h
Name	Number Of Analog Input Offsets
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	01_h

Subindex	01_h
Name	Analog Input #1 Offset
Data type	INTEGER16
Access	read / write
PDO mapping	no
Allowed values	
Preset value	0000_h

3322h Analog Input Numerators

Function

Value by which the read analog value (3220_h, 3321_h) is multiplied before it is written in object 3320_h.

Object description

Index	3322_h
Object name	Analog Input Numerators
Object Code	ARRAY
Data type	INTEGER16
Savable	yes, category: application
Access	read only
PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v1426
Change history	

Value description

Subindex	00_h
Name	Number Of Analog Input Numerators
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	01_h

Subindex	01_h
Name	Analog Input #1 Numerator
Data type	INTEGER16
Access	read / write
PDO mapping	no
Allowed values	
Preset value	7850_h

Description

The subindices contain:

- Subindex 01_h: Multiplier for analog input 1
- Subindex 02_h: Multiplier for analog input 2 (if present)

3323h Analog Input Denominators

Function

Value by which the read analog value (3220_h+ 3321_h) is divided before it is written in object 3320_h.

Object description

Index	3323_h
Object name	Analog Input Denominators
Object Code	ARRAY
Data type	INTEGER16
Savable	yes, category: application
Access	read only
PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v1926-B648637
Change history	

Value description

Subindex	00_h

Name	Number Of Analog Input Denominators
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	01_h

Subindex	01_h
Name	Analog Input #1 Denominator
Data type	INTEGER16
Access	read / write
PDO mapping	no
Allowed values	
Preset value	$0FFF_h$

Description

The subindices contain:

- Subindex 01_h: Divisor for analog input 1
- Subindex 02_h: Divisor for analog input 2 (if present)

3380h Feedback Sensorless

Function

Contains measurement and configuration values that are necessary for the sensorless control and field weakening in Closed-Loop.

Object description

Index	3380_h
Object name	Feedback Sensorless
Object Code	ARRAY
Data type	UNSIGNED32
Savable	yes, category: tuning
Access	read only
PDO mapping	RX-PDO
Allowed values	
Preset value	
Firmware version	FIR-v2013-B726332
Change history	Firmware version FIR-v2013-B726332: The number of entries was changed from 7 to 6.

Value description

Subindex	00_h
Name	Number Of Entries

Data type	UNSIGNED8
Access	read only
PDO mapping	RX-PDO
Allowed values	
Preset value	05_h

Subindex	01_h
Name	Resistance [Ohm]
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	■ PD1-C281S15-E-20-2: $3FA66666_h$ ■ PD1-C281S15-E-20-5: $3FA66666_h$ ■ PD1-C281S15-E-65-2: $3FA66666_h$ ■ PD1-C281S15-E-65-5: $3FA66666_h$ ■ PD1-C281S15-E-OF-2: $3FA66666_h$ ■ PD1-C281S15-E-OF-5: $3FA66666_h$ ■ PD1-C281L15-E-20-2: $3FF33333_h$ ■ PD1-C281L15-E-20-5: $3FF33333_h$ ■ PD1-C281L15-E-65-2: $3FF33333_h$ ■ PD1-C281L15-E-65-5: $3FF33333_h$ ■ PD1-C281L15-E-OF-2: $3FF33333_h$ ■ PD1-C281L15-E-OF-5: $3FF33333_h$

Subindex	02_h
Name	Inductance [H]
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	■ PD1-C281S15-E-20-2: $3A83126F_h$ ■ PD1-C281S15-E-20-5: $3A83126F_h$ ■ PD1-C281S15-E-65-2: $3A83126F_h$ ■ PD1-C281S15-E-65-5: $3A83126F_h$ ■ PD1-C281S15-E-OF-2: $3A83126F_h$ ■ PD1-C281S15-E-OF-5: $3A83126F_h$ ■ PD1-C281L15-E-20-2: $3AF9096C_h$ ■ PD1-C281L15-E-20-5: $3AF9096C_h$ ■ PD1-C281L15-E-65-2: $3AF9096C_h$ ■ PD1-C281L15-E-65-5: $3AF9096C_h$ ■ PD1-C281L15-E-OF-2: $3AF9096C_h$ ■ PD1-C281L15-E-OF-5: $3AF9096C_h$

Subindex	03_h
Name	Magnetic Flux [Vs]

Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	■ PD1-C281S15-E-20-2: 3A5E6FE2$_h$ ■ PD1-C281S15-E-20-5: 3A5E6FE2$_h$ ■ PD1-C281S15-E-65-2: 3A5E6FE2$_h$ ■ PD1-C281S15-E-65-5: 3A5E6FE2$_h$ ■ PD1-C281S15-E-OF-2: 3A5E6FE2$_h$ ■ PD1-C281S15-E-OF-5: 3A5E6FE2$_h$ ■ PD1-C281L15-E-20-2: 3ADE6FE2$_h$ ■ PD1-C281L15-E-20-5: 3ADE6FE2$_h$ ■ PD1-C281L15-E-65-2: 3ADE6FE2$_h$ ■ PD1-C281L15-E-65-5: 3ADE6FE2$_h$ ■ PD1-C281L15-E-OF-2: 3ADE6FE2$_h$ ■ PD1-C281L15-E-OF-5: 3ADE6FE2$_h$

Subindex	04$_h$
Name	Switch On Speed [rpm]
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000078$_h$

Subindex	05$_h$
Name	Switch Off Speed [rpm]
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000064$_h$

Description

The subindices have the following function:

- 01$_h$: Winding resistance. Float value, shown here as UNSIGNED32. Is determined by Auto setup.
- 02$_h$: Winding inductance. Float value, shown here as UNSIGNED32. Is determined by Auto setup.
- 03$_h$: Interlinking flux. Float value, shown here as UNSIGNED32. Is determined by Auto setup.
- 04$_h$: Switch-on speed in RPM. *Closed loop* (*sensorless*) is activated above this speed if no sensors were detected by Auto setup.
- 05$_h$: Switch-off speed in RPM. *Closed loop* (*sensorless*) is deactivated below this speed if no sensors were detected by Auto setup.

33A0h Feedback Incremental A/B/I 1

Function

Contains configuration values for the first incremental encoder. The values are determined by the Auto setup.

Object description

Index	$33A0_h$
Object name	Feedback Incremental A/B/I 1
Object Code	ARRAY
Data type	UNSIGNED16
Savable	yes, category: tuning
Access	read only
PDO mapping	RX-PDO
Allowed values	
Preset value	
Firmware version	FIR-v1738-B501312
Change history	

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	RX-PDO
Allowed values	
Preset value	03_h

Subindex	01_h
Name	Configuration
Data type	UNSIGNED16
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	0001_h

Subindex	02_h
Name	Alignment
Data type	UNSIGNED16
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	0000_h

Subindex	03_h
Name	Latency [ns]
Data type	UNSIGNED16
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	0000_h

Description

The subindices have the following function:

- 01_h (Configuration): The following bits have a meaning:
  - Bit 0: Value = "0": The encoder does not have an index. Value = "1": Encoder index exists and is to be used.
- 02_h (Alignment): This value specifies the offset between the index of the encoder and the rotor's magnets. The exact determination is possible via auto setup. The presence of this value is necessary for *closed-loop* mode with encoder.
- 03_h (Latency): Here you can enter the latency of used encoder.

3502h MODBUS Rx PDO Mapping

Function

The objects for RX mapping can be written in this object.

NOTICE

To be able to change the mapping, you must first deactivate it by setting subindex 0_h to "0".

After writing the objects to the respective subindices, enter the number of mapped objects in subindex 0_h.

Object description

Index	3502_h
Object name	MODBUS Rx PDO Mapping
Object Code	ARRAY
Data type	UNSIGNED32
Savable	yes, category: communication
Access	read / write
PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v1748-B538662
Change history	Firmware version FIR-v1738-B505321: "Object Name" entry changed from "MODBUS Rx PDO-Mapping" to "MODBUS Rx PDO Mapping".

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read / write
PDO mapping	no
Allowed values	
Preset value	07_h

Subindex	01_h
Name	1st Object To Be Mapped
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	60400010_h

Subindex	02_h
Name	2nd Object To Be Mapped
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00050008_h

Subindex	03_h
Name	3rd Object To Be Mapped
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	60600008_h

Subindex	04_h
Name	4th Object To Be Mapped
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	$607A0020_h$

Subindex	05_h

Name	5th Object To Be Mapped
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	60810020_h

Subindex	06_h
Name	6th Object To Be Mapped
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	60420010_h

Subindex	07_h
Name	7th Object To Be Mapped
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	$60FE0120_h$

Subindex	08_h
Name	8th Object To Be Mapped
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000_h

Subindex	09_h
Name	9th Object To Be Mapped
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000_h

Subindex	$0A_h$
Name	10th Object To Be Mapped
Data type	UNSIGNED32
Access	read / write

PDO mapping	no
Allowed values	
Preset value	00000000_h

Subindex	$0B_h$
Name	11th Object To Be Mapped
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000_h

Subindex	$0C_h$
Name	12th Object To Be Mapped
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000_h

Subindex	$0D_h$
Name	13th Object To Be Mapped
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000_h

Subindex	$0E_h$
Name	14th Object To Be Mapped
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000_h

Subindex	$0F_h$
Name	15th Object To Be Mapped
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000_h

Subindex	10_h
Name	16th Object To Be Mapped
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000_h

3602h MODBUS Tx PDO Mapping

Function

The objects for TX mapping can be written in this object.

NOTICE

To be able to change the mapping, you must first deactivate it by setting subindex 0_h to "0".

After writing the objects to the respective subindices, enter the number of mapped objects in subindex 0_h.

Object description

Index	3602_h
Object name	MODBUS Tx PDO Mapping
Object Code	ARRAY
Data type	UNSIGNED32
Savable	yes, category: communication
Access	read / write
PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v1748-B538662
Change history	Firmware version FIR-v1738-B505321: "Object Name" entry changed from "MODBUS Tx PDO-Mapping" to "MODBUS Tx PDO Mapping".

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read / write
PDO mapping	no
Allowed values	
Preset value	06_h

Subindex	01_h
Name	1st Object To Be Mapped
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	60410010_h

Subindex	02_h
Name	2nd Object To Be Mapped
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00050008_h

Subindex	03_h
Name	3rd Object To Be Mapped
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	60610008_h

Subindex	04_h
Name	4th Object To Be Mapped
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	60640020_h

Subindex	05_h
Name	5th Object To Be Mapped
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	60440010_h

Subindex	06_h
Name	6th Object To Be Mapped
Data type	UNSIGNED32

Access	read / write
PDO mapping	no
Allowed values	
Preset value	$60FD0020_h$

Subindex	07_h
Name	7th Object To Be Mapped
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000_h

Subindex	08_h
Name	8th Object To Be Mapped
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000_h

Subindex	09_h
Name	9th Object To Be Mapped
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000_h

Subindex	$0A_h$
Name	10th Object To Be Mapped
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000_h

Subindex	$0B_h$
Name	11th Object To Be Mapped
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	

Preset value	00000000_h

Subindex	$0C_h$
Name	12th Object To Be Mapped
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000_h

Subindex	$0D_h$
Name	13th Object To Be Mapped
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000_h

Subindex	$0E_h$
Name	14th Object To Be Mapped
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000_h

Subindex	$0F_h$
Name	15th Object To Be Mapped
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000_h

Subindex	10_h
Name	16th Object To Be Mapped
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000_h

3700h Deviation Error Option Code

Function

The object contains the action that is to be executed if a following or slippage error is triggered.

Object description

Index	3700_h
Object name	Deviation Error Option Code
Object Code	VARIABLE
Data type	INTEGER16
Savable	yes, category: application
Access	read / write
PDO mapping	no
Allowed values	
Preset value	$FFFF_h$
Firmware version	FIR-v1426
Change history	Firmware version FIR-v1738-B501312: "Object Name" entry changed from "Following Error Option Code" to "Deviation Error Option Code".

Description

Value	Description
-32768 … -2	Reserved
-1	no reaction
0	Switch off driver without deceleration ramp; drive function blocked – motor can turn freely
1	Braking with *slow down ramp* (braking deceleration depending on operating mode)
2	Braking with *quick stop ramp* (6085_h)
3 … 32767	reserved

3701h Limit Switch Error Option Code

Function

If a limit switch is triggered, the limit switch position is stored internally, bit 7 (*Warning*) in 6041_h (*statusword*) is set and the CiA 402 Power State Machine is set to the *Quick Stop Active* state. The action stored in this object is executed in the process. See chapter Limitation of the range of motion.

Object description

Index	3701_h
Object name	Limit Switch Error Option Code
Object Code	VARIABLE
Data type	INTEGER16
Savable	yes, category: application

Access	read / write
PDO mapping	no
Allowed values	
Preset value	$FFFF_h$
Firmware version	FIR-v1748-B538662
Change history	

Description

Value in object 3701_h	Description
-2	No reaction, discard the limit switch position
-1 (factory settings)	No reaction (e. g., to execute a homing operation) except noting the limit switch position
0	Switch off driver without deceleration ramp; drive function blocked – motor can turn freely (*Switch on disabled* state)
1	Braking with *slow down ramp* (deceleration ramp depending on operating mode) and subsequent state change to *Switch on disabled*
2	Braking with *quick stop ramp* and subsequent state change to *Switch on disabled*
5	Braking with *slow down ramp* (deceleration ramp depending on operating mode) and subsequent state change to *Quick stop active*; control does not switch off and the motor remains energized. You can switch back to the *Operation enabled* state.
6	Braking with *quick stop ramp* and subsequent state change to *Quick Stop Active*; control does not switch off and the motor remains energized. You can switch back to the *Operation enabled* state.

4012h HW Information

Function

This object contains information about the hardware.

Object description

Index	4012_h
Object name	HW Information
Object Code	ARRAY
Data type	UNSIGNED32
Savable	no
Access	read only
PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v1540
Change history	

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	01_h

Subindex	01_h
Name	EEPROM Size In Bytes
Data type	UNSIGNED32
Access	read only
PDO mapping	no
Allowed values	
Preset value	00000000_h

Description

Subindex 01: Contains the size of the connected EEPROM in bytes. The value "0" means that no EEPROM is connected.

4013h HW Configuration

Function

This object is used to set certain hardware configurations.

Object description

Index	4013_h
Object name	HW Configuration
Object Code	ARRAY
Data type	UNSIGNED32
Savable	yes, category: application
Access	read only
PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v1540
Change history	

Value description

Subindex	00_h
Name	Number Of Entries

Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	01_h

Subindex	01_h
Name	HW Configuration #1
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000_h

Description

4014h Operating Conditions

Function

This object is used to read out the current environment values for the controller.

Object description

Index	4014_h
Object name	Operating Conditions
Object Code	ARRAY
Data type	INTEGER32
Savable	no
Access	read only
PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v1540
Change history	Firmware version FIR-v1650-B472161: "Access" table entry for subindex 01 changed from "read/write" to "read only". Firmware version FIR-v1650-B472161: "Access" table entry for subindex 02 changed from "read/write" to "read only". Firmware version FIR-v1650-B472161: "Name" entry changed from "Temperature PCB [d?C]" to "Temperature PCB [Celsius * 10]". Firmware version FIR-v1650-B472161: "Access" table entry for subindex 03 changed from "read/write" to "read only". Firmware version FIR-v1738-B501312: The number of entries was changed from 4 to 6.

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	05_h

Subindex	01_h
Name	Voltage UB Power [mV]
Data type	INTEGER32
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	00000000_h

Subindex	02_h
Name	Voltage UB Logic [mV]
Data type	INTEGER32
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	00000000_h

Subindex	03_h
Name	Temperature PCB [Celsius * 10]
Data type	INTEGER32
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	00000000_h

Subindex	04_h
Name	Temperature Motor [Celsius * 10]
Data type	INTEGER32
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	00000000_h

Subindex	05_h

Name	Temperature Microcontroller Chip [Celsius * 10]
Data type	INTEGER32
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	00000000_h

Description

The subindices contain:

- 01_h: Current voltage supply voltage in [mV]
- 02_h:
- 03_h: Current temperature of the control board in [d°C] (tenths of degree)
- 04_h: Reserves
- 05_h: Current temperature of the processor in [d°C] (tenths of degree)

4015h Special Drive Modes

Function

With this object, you can switch the *special drive modes* off or on. See chapter .

Object description

Index	4015_h
Object name	Special Drive Modes
Object Code	ARRAY
Data type	UNSIGNED8
Savable	yes, category: application
Access	read only
PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v1738-B501312
Change history	

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	02_h

Subindex	01_h

Name	Special Drive Mode Configuration
Data type	UNSIGNED8
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00_h

Subindex	02_h
Name	Virtual Config Switch Value
Data type	UNSIGNED8
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00_h

Description

The subindices have the following functions:

- 01_h:
  - Value ="0"$_h$: The *special drive modes* are switched off
  - Value="2"$_h$: The *special drive modes* are switched on and the mode is set in subindex 02_h.
- 02_h: Defines the used mode.

4016h Factory Setup

Function

This object indicates whether Auto setup was executed in production.

Object description

Index	4016_h
Object name	Factory Setup
Object Code	ARRAY
Data type	UNSIGNED32
Savable	no
Access	read only
PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v1738-B501312
Change history	Firmware version FIR-v2013-B726332: "Access" table entry for subindex 01 changed from "read/write" to "read only".

Value description

Subindex	00_h

Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	02_h

Subindex	01_h
Name	Factory Setup Successful
Data type	UNSIGNED32
Access	read only
PDO mapping	no
Allowed values	
Preset value	00000000_h

Subindex	02_h
Name	Factory Setup Status
Data type	UNSIGNED32
Access	read only
PDO mapping	no
Allowed values	
Preset value	00000000_h

Description

Valid values for subindex 01_h:

- Value = "0": *Auto setup* was not executed.
- Value = "1": *Auto setup* was executed.

4040h Drive Serial Number

Function

This object contains the serial number of the controller.

Object description

Index	4040_h
Object name	Drive Serial Number
Object Code	VARIABLE
Data type	VISIBLE_STRING
Savable	no
Access	read only
PDO mapping	no
Allowed values	
Preset value	0
Firmware version	FIR-v1450

Change history	

4041h Device Id

Function

This object contains the ID of the device.

Object description

Index	4041_h
Object name	Device Id
Object Code	VARIABLE
Data type	OCTET_STRING
Savable	no
Access	read only
PDO mapping	no
Allowed values	
Preset value	0
Firmware version	FIR-v1540
Change history	

4042h Bootloader Infos

Object description

Index	4042_h
Object name	Bootloader Infos
Object Code	ARRAY
Data type	UNSIGNED32
Savable	no
Access	read only
PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v2013-B726332
Change history	

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	

Preset value	03_h

Subindex	01_h
Name	Bootloader Version
Data type	UNSIGNED32
Access	read only
PDO mapping	no
Allowed values	
Preset value	00000000_h

Subindex	02_h
Name	Bootloader Supported Fieldbus
Data type	UNSIGNED32
Access	read only
PDO mapping	no
Allowed values	
Preset value	00000000_h

Subindex	03_h
Name	Bootloader Hw-group
Data type	UNSIGNED32
Access	read only
PDO mapping	no
Allowed values	
Preset value	00000000_h

Description

The subindices have the following functions:

- 01_h: Version of the boot loader. The 4 most significant bytes contain the main version number; the 4 least significant bytes contain the minor version number. Example for version 4.2: 00040002_h
- 02_h: Fieldbuses supported by the boot loader. The bits have the same function as the bits of object 2101h Fieldbus Module Availability.

6007h Abort Connection Option Code

Function

If an error (watchdog, heartbeat, etc.) occurs on the bus, the controller automatically switches to the SAFEOPERATIONAL state. With this object, you can set the reaction.

Object description

Index	6007_h
Object name	Abort Connection Option Code
Object Code	VARIABLE
Data type	INTEGER16

Savable	yes, category: application
Access	read / write
PDO mapping	no
Allowed values	
Preset value	0001_h
Firmware version	FIR-v2013-B726332
Change history	

Description

You can set the following reactions:

Value	Reaction
-1	The controller (slave) sets all input values (RX-PDO) to the value "0". This also sets the object 6040_h (controlword) to "0", which causes the motor to coast to a stop.
0	no reaction
1	*Fault*: The action stored in object $605E_h$ is executed.
2	*Disable voltage*: Transition to the *Switched on disabled* state without halt motion reaction (the motor coasts to a stop)
3	*Quick stop*: The action stored in object $605A_h$ is executed.

603Fh Error Code

Function

This object returns the error code of the last error that occurred.

It corresponds to the lower 16 bits of object 1003_h. For the description of the error codes, refer to object 1003_h.

Object description

Index	$603F_h$
Object name	Error Code
Object Code	VARIABLE
Data type	UNSIGNED16
Savable	no
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	0000_h
Firmware version	FIR-v1426
Change history	

Description

For the meaning of the error, see object 1003_h (Pre-defined Error Field).

If the error is reset by setting bit 7 in 6040h Controlword, this object is also automatically reset to "0".

6040h Controlword

Function

This object controls the CiA 402 Power State Machine.

Object description

Index	6040_h
Object name	Controlword
Object Code	VARIABLE
Data type	UNSIGNED16
Savable	yes, category: application
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	0000_h
Firmware version	FIR-v1426
Change history	Firmware version FIR-v1626: "Savable" entry changed from "no" to "yes, category: application".

Description

Parts of the object are, with respect to function, dependent on the currently selected mode.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
						OMS	HALT	FR		OMS [3]		EO	QS	EV	SO

SO (Switched On)

Value = "1": Switches to the "Switched on" state

EV (Enable Voltage)

Value = "1": Switches to the "Enable voltage" state

QS (Quick Stop)

Value = "0": Switches to the "Quick stop" state

EO (Enable Operation)

Value = "1": Switches to the "Enable operation" state

OMS (Operation Mode Specific)

Meaning is dependent on the selected operating mode

FR (Fault Reset)

Resets an error or a warning (if possible)

HALT

Value = "1": Triggers a halt; valid in the following modes:

- Profile Position
- Velocity
- Profile Velocity
- Profile Torque

■ Interpolated Position Mode

6041h Statusword

Function

This object returns information about the status of the CiA 402 Power State Machine.

Object description

Index	6041_h
Object name	Statusword
Object Code	VARIABLE
Data type	UNSIGNED16
Savable	no
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	0000_h
Firmware version	FIR-v1426
Change history	

Description

Parts of the object are, with respect to function, dependent on the currently selected mode. Refer to the corresponding section in chapter operating modes.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CLA		OMS [2]		ILA	TARG	REM	SYNC	WARN	SOD	QS	VE	FAULT	OE	SO	RTSO

RTSO (Ready To Switch On)

Value = "1": Controller is in the "Ready to switch on" state

SO (Switched On)

Value = "1": Controller is in the "Switched on" state

OE (Operation Enabled)

Value = "1": Controller is in the "Operation enabled" state

FAULT

Error occurred (see 1003_h)

VE (Voltage Enabled)

Voltage applied

QS (Quick Stop)

Value = "0": Controller is in the "Quick stop" state

SOD (Switched On Disabled)

Value = "1": Controller is in the "Switched on disabled" state

WARN (Warning)

Value = "1": Warning

SYNC (synchronization)
Value = "1": Controller is in sync with the fieldbus; value = "0": Controller is not in sync with the fieldbus

REM (Remote)
Remote (value of the bit is always "1")

TARG
Target reached. The value stays after an operation mode change at "1", until a new target is commanded.

ILA (Internal Limit Active)
Limit exceeded

OMS (Operation Mode Specific)
Meaning is dependent on the selected operating mode

CLA (Closed Loop Active)
Value = "1": The controller is in the *Operation enabled* state and the Closed-Loop is activated.

Listed in the following table are the bit masks that break down the state of the controller.

Statusword (6041_h)	State
xxxx xxxx x0xx 0000	Not ready to switch on
xxxx xxxx x1xx 0000	Switch on disabled
xxxx xxxx x01x 0001	Ready to switch on
xxxx xxxx x01x 0011	Switched on
xxxx xxxx x01x 0111	Operation enabled
xxxx xxxx x00x 0111	Quick stop active
xxxx xxxx x0xx 1111	Fault reaction active
xxxx xxxx x0xx 1000	Fault

6042h Vl Target Velocity

Function

Specifies the target speed in user-defined units for Velocity mode.

Object description

Index	6042_h
Object name	Vl Target Velocity
Object Code	VARIABLE
Data type	INTEGER16
Savable	yes, category: application
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	0000_h
Firmware version	FIR-v1426

Change history	Firmware version FIR-v1626: "Savable" entry changed from "no" to "yes, category: application".

6043h Vl Velocity Demand

Function

Speed specification in user-defined units for the controller in Velocity mode.

Object description

Index	6043_h
Object name	Vl Velocity Demand
Object Code	VARIABLE
Data type	INTEGER16
Savable	no
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	0000_h
Firmware version	FIR-v1426
Change history	

6044h Vl Velocity Actual Value

Function

Specifies the current actual speed in user-defined units in Velocity mode.

Object description

Index	6044_h
Object name	Vl Velocity Actual Value
Object Code	VARIABLE
Data type	INTEGER16
Savable	no
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	0000_h
Firmware version	FIR-v1426
Change history	

6046h Vl Velocity Min Max Amount

Function

This object can be used to set the minimum speed and maximum speed in user-defined units.

Object description

Index	6046_h
Object name	Vl Velocity Min Max Amount
Object Code	ARRAY
Data type	UNSIGNED32
Savable	yes, category: application
Firmware version	FIR-v1426
Change history	

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	02_h

Subindex	01_h
Name	Min Amount
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000000_h

Subindex	02_h
Name	Max Amount
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00007530_h

Description

Subindex 1 contains the minimum speed.

Subindex 2 contains the maximum speed.

If the value of the target speed (object 6042_h) specified here is less than the minimum speed, the minimum speed applies and bit 11 (Internal Limit Reached) in 6041h Statusword$_h$ is set.

A target speed greater than the maximum speed sets the speed to the maximum speed and bit 11 (Internal Limit Reached) in 6041h Statusword$_h$ is set.

6048h Vl Velocity Acceleration

Function

Sets the acceleration ramp in Velocity Mode (see Velocity).

Object description

Index	6048_h
Object name	Vl Velocity Acceleration
Object Code	RECORD
Data type	VELOCITY_ACCELERATION_DECELERATION
Savable	yes, category: application
Firmware version	FIR-v1426
Change history	

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	02_h

Subindex	01_h
Name	Delta Speed
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	$000001F4_h$

Subindex	02_h
Name	Delta Time
Data type	UNSIGNED16
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	0001_h

Description

The acceleration is specified as a fraction in user-defined units:

Speed change per change in time.

Subindex 01_h: Contains the change in speed.

Subindex 02_h: Contains the change in time.

6049h Vl Velocity Deceleration

Function

Sets the deceleration (deceleration ramp) in Velocity Mode (see Velocity).

Object description

Index	6049_h
Object name	Vl Velocity Deceleration
Object Code	RECORD
Data type	VELOCITY_ACCELERATION_DECELERATION
Savable	yes, category: application
Firmware version	FIR-v1426
Change history	

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	02_h

Subindex	01_h
Name	Delta Speed
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	$000001F4_h$

Subindex	02_h
Name	Delta Time
Data type	UNSIGNED16
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	0001_h

Description

The deceleration is specified as a fraction in user-defined units:

Speed change per change in time.

Subindex 01_h: Contains the change in speed.

Subindex 02_h: Contains the change in time.

604Ah Vl Velocity Quick Stop

Function

This object defines the deceleration (deceleration ramp) if the Quick Stop state is initiated in velocity mode.

Object description

Index	$604A_h$
Object name	Vl Velocity Quick Stop
Object Code	RECORD
Data type	VELOCITY_ACCELERATION_DECELERATION
Savable	yes, category: application
Firmware version	FIR-v1426
Change history	

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	02_h

Subindex	01_h
Name	Delta Speed
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00001388_h

Subindex	02_h
Name	Delta Time
Data type	UNSIGNED16
Access	read / write
PDO mapping	RX-PDO
Allowed values	

Preset value	0001_h

Description

The deceleration is specified as a fraction in user-defined units:

Speed change per change in time.

Subindex 01_h: Contains the change in speed.

Subindex 02_h: Contains the change in time.

604Ch Vl Dimension Factor

Function

The unit for speed values is defined here for the objects associated with velocity mode.

Object description

Index	$604C_h$
Object name	Vl Dimension Factor
Object Code	ARRAY
Data type	INTEGER32
Savable	yes, category: application
Firmware version	FIR-v1426
Change history	

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	02_h

Subindex	01_h
Name	Vl Dimension Factor Numerator
Data type	INTEGER32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000001_h

Subindex	02_h
Name	Vl Dimension Factor Denominator
Data type	INTEGER32

Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000001_h

Description

Subindex 1 contains the numerator (multiplier) and subindex 2 contains the denominator (divisor) with which the internal speed values are converted to revolutions per minute. If, for example, subindex 1 is set to the value "60" and subindex 2 is set to the value "1", the speed is specified in revolutions per second (60 revolutions per 1 minute).

605Ah Quick Stop Option Code

Function

The object contains the action that is to be executed on a transition of the CiA 402 Power State Machine to the *Quick Stop active* state.

Object description

Index	$605A_h$
Object name	Quick Stop Option Code
Object Code	VARIABLE
Data type	INTEGER16
Savable	yes, category: application
Access	read / write
PDO mapping	no
Allowed values	
Preset value	0002_h
Firmware version	FIR-v1426
Change history	

Description

Value in object $605A_h$	Description
0	Immediate stop with subsequent state change to *Switch on disabled*
1	Braking with *slow down ramp* (deceleration ramp depending on operating mode) and subsequent state change to *Switch on disabled*
2	Braking with *quick stop ramp* (6085_h) and subsequent state change to *Switch on disabled*
5	Braking with *slow down ramp* (deceleration ramp depending on operating mode) and subsequent state change to *Quick stop active*; control does not switch off and the motor remains energized. You can switch back to the *Operation enabled* state.
6	Braking with *quick stop ramp* (6085_h) and subsequent state change to *Quick Stop Active*; control does not switch off and the motor remains energized. You can switch back to the *Operation enabled* state.

In the mode Homing, the deceleration ramp set in $609A_h$ (Homing Acceleration) is used, instead of 6085_h.

605Bh Shutdown Option Code

Function

This object contains the action that is to be executed on a transition of the CiA 402 Power State Machine from the *Operation enabled* state to the *Ready to switch on* state.

Object description

Index	$605B_h$
Object name	Shutdown Option Code
Object Code	VARIABLE
Data type	INTEGER16
Savable	yes, category: application
Access	read / write
PDO mapping	no
Allowed values	
Preset value	0000_h
Firmware version	FIR-v1426
Change history	

Description

Value in object $605B_h$	Description
-32768 … -1	Reserved
0	Blocking of the drive function – motor can turn freely
1	Braking with *slow down ramp* (braking deceleration depending on operating mode) and subsequent state change to *Ready to switch on*
2 … 32767	Reserved

605Ch Disable Operation Option Code

Function

This object contains the action that is to be executed on a transition of the CiA 402 Power State Machine from the *Operation enabled* state to the *Switched on* state.

Object description

Index	$605C_h$
Object name	Disable Operation Option Code
Object Code	VARIABLE
Data type	INTEGER16
Savable	yes, category: application
Access	read / write
PDO mapping	no
Allowed values	

Preset value	0001_h
Firmware version	FIR-v1426
Change history	Firmware version FIR-v2412-B1057638: entry "Object Name" changed from "Disable Option Code" to "Disable Operation Option Code".

Description

Value in object $605C_h$	Description
-32768 … -1	Reserved
0	Switch off driver without deceleration ramp; drive function blocked
1	Braking with *slow down ramp* (braking deceleration depending on operating mode) and subsequent state change to *Switched on*
2 … 32767	Reserved

605Dh Halt Option Code

Function

The object contains the action that is to be executed if bit 8 (Halt) is set in controlword 6040_h.

Object description

Index	$605D_h$
Object name	Halt Option Code
Object Code	VARIABLE
Data type	INTEGER16
Savable	yes, category: application
Access	read / write
PDO mapping	no
Allowed values	
Preset value	0001_h
Firmware version	FIR-v1426
Change history	

Description

Value in object $605D_h$	Description
-32768 … 0	Reserved
1	Braking with *slow down ramp* (braking deceleration depending on operating mode)
2	Braking with *quick stop ramp* (6085_h)
3 … 32767	Reserved

605Eh Fault Reaction Option Code

Function

The object contains the action specifying how the motor is to be brought to a standstill in case of an error.

Object description

Index	$605E_h$
Object name	Fault Reaction Option Code
Object Code	VARIABLE
Data type	INTEGER16
Savable	yes, category: application
Access	read / write
PDO mapping	no
Allowed values	
Preset value	0002_h
Firmware version	FIR-v1426
Change history	Firmware version FIR-v2412-B1057638: entry "Object Name" changed from "Fault Option Code" to "Fault Reaction Option Code".

Description

Value in object $605E_h$	Description
-32768 … -1	Reserved
0	Blocking of the drive function – motor can turn freely
1	Braking with *slow down ramp* (braking deceleration depending on operating mode)
2	Braking with *quick stop ramp* (6085_h)
3 … 32767	Reserved

6060h Modes Of Operation

Function

The desired operating mode is entered in this object.

Object description

Index	6060_h
Object name	Modes Of Operation
Object Code	VARIABLE
Data type	INTEGER8
Savable	yes, category: application
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00_h

Firmware version	FIR-v1426
Change history	Firmware version FIR-v1626: "Savable" entry changed from "no" to "yes, category: application".

Description

Mode	Description
0	No mode change/no mode assigned
1	Profile Position Mode
2	Velocity Mode
3	Profile Velocity Mode
4	Profile Torque Mode
5	Reserved
6	Homing Mode

6061h Modes Of Operation Display

Function

Indicates the current operating mode. See also 6060h Modes Of Operation.

Object description

Index	6061_h
Object name	Modes Of Operation Display
Object Code	VARIABLE
Data type	INTEGER8
Savable	no
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	00_h
Firmware version	FIR-v1426
Change history	

6062h Position Demand Value

Function

Indicates the current demand position in user-defined units.

Object description

Index	6062_h
Object name	Position Demand Value
Object Code	VARIABLE
Data type	INTEGER32

Savable	no
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	00000000_h
Firmware version	FIR-v1426
Change history	

6063h Position Actual Internal Value

Function

Contains the current actual position in encoder increments. Unlike objects 6062_h and 6064_h, this value is not set to "0" following a Homing operation. The source is determined in 3203h Feedback Selection.

NOTICE

If the encoder resolution in object $608F_h$ = zero, the numerical values of this object are invalid.

Object description

Index	6063_h
Object name	Position Actual Internal Value
Object Code	VARIABLE
Data type	INTEGER32
Savable	no
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	00000000_h
Firmware version	FIR-v1426
Change history	

6064h Position Actual Value

Function

Contains the current actual position in user-defined units. The source is determined in 3203h Feedback Selection.

Object description

Index	6064_h
Object name	Position Actual Value
Object Code	VARIABLE
Data type	INTEGER32
Savable	no
Access	read only

PDO mapping	TX-PDO
Allowed values	
Preset value	00000000_h
Firmware version	FIR-v1426
Change history	

6065h Following Error Window

Function

Defines the maximum allowed following error in user-defined units symmetrically to the demand position.

Object description

Index	6065_h
Object name	Following Error Window
Object Code	VARIABLE
Data type	UNSIGNED32
Savable	yes, category: application
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000100_h
Firmware version	FIR-v1426
Change history	Firmware version FIR-v1504: "Savable" entry changed from "no" to "yes, category: application".

Description

If the actual position deviates so much from the demand position that the value of this object is exceeded, bit 13 in object 6041_h is set. The deviation must last longer than the time in object 6066_h.

If the value of the "Following Error Window" is set to "FFFFFFFF"$_h$, following error monitoring is switched off.

A reaction to the following error can be set in object 3700_h. If a reaction is defined, an error is also entered in object 1003_h.

6066h Following Error Time Out

Function

Time in milliseconds until a larger following error results in an error message.

Object description

Index	6066_h
Object name	Following Error Time Out
Object Code	VARIABLE
Data type	UNSIGNED16
Savable	yes, category: application
Access	read / write

PDO mapping	RX-PDO
Allowed values	
Preset value	0064_h
Firmware version	FIR-v1426
Change history	Firmware version FIR-v1504: "Savable" entry changed from "no" to "yes, category: application".

Description

If the actual position deviates so much from the demand position that the value of object 6065_h is exceeded, bit 13 in object 6041_h is set. The deviation must persist for longer than the time defined in this object.

A reaction to the following error can be set in object 3700_h. If a reaction is defined, an error is also entered in object 1003_h.

6067h Position Window

Function

Specifies a range symmetrical to the target position within which that target is considered having been met in modes Profile Position and Interpolated Position Mode.

Object description

Index	6067_h
Object name	Position Window
Object Code	VARIABLE
Data type	UNSIGNED32
Savable	yes, category: application
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	$0000000A_h$
Firmware version	FIR-v1426
Change history	Firmware version FIR-v1504: "Savable" entry changed from "no" to "yes, category: application".

Description

If the current position deviates from the target position by less than the value of this object, bit 10 in object 6041_h is set. The condition must be satisfied for longer than the time defined in object 6068_h.

If the value is set to "FFFFFFFF"$_h$, monitoring is switched off.

6068h Position Window Time

Function

The current position must be within the "Position Window" (6067_h) for this time in milliseconds for the target position to be considered having been met in the Profile Position and Interpolated Position Mode modes.

Object description

Index	6068_h
Object name	Position Window Time
Object Code	VARIABLE
Data type	UNSIGNED16
Savable	yes, category: application
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	0064_h
Firmware version	FIR-v1426
Change history	Firmware version FIR-v1504: "Savable" entry changed from "no" to "yes, category: application".

Description

If the current position deviates from the target position by less than the value of object 6067_h, bit 10 in object 6041_h is set. The condition must be satisfied for longer than the time defined in object 6068_h.

606Bh Velocity Demand Value

Function

Speed specification in user-defined units for the velocity controller.

Object description

Index	$606B_h$
Object name	Velocity Demand Value
Object Code	VARIABLE
Data type	INTEGER32
Savable	no
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	00000000_h
Firmware version	FIR-v1426
Change history	

Description

This object contains the output of the ramp generator, which simultaneously serves as the preset value for the velocity controller.

606Ch Velocity Actual Value

Function

Current actual speed in user-defined units.

Object description

Index	$606C_h$
Object name	Velocity Actual Value
Object Code	VARIABLE
Data type	INTEGER32
Savable	no
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	00000000_h
Firmware version	FIR-v1426
Change history	

606Dh Velocity Window

Function

Specifies a symmetrical range relative to the target speed within which the target is considered having been met in the Profile Velocity mode.

Object description

Index	$606D_h$
Object name	Velocity Window
Object Code	VARIABLE
Data type	UNSIGNED16
Savable	yes, category: application
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	$001E_h$
Firmware version	FIR-v1426
Change history	Firmware version FIR-v1614: "Savable" entry changed from "no" to "yes, category: application".

Description

If the current speed deviates from the set speed by less than the value of this object, bit 10 in object 6041_h is set. The condition must be satisfied for longer than the time defined in object $606E_h$ (see also statusword in Profile Velocity Mode).

606Eh Velocity Window Time

Function

The current speed must be within the "Velocity Window" ($606D_h$) for this time (in milliseconds) for the target to be considered having been met.

Object description

Index	606E$_h$
Object name	Velocity Window Time
Object Code	VARIABLE
Data type	UNSIGNED16
Savable	yes, category: application
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	0000$_h$
Firmware version	FIR-v1426
Change history	Firmware version FIR-v1614: "Savable" entry changed from "no" to "yes, category: application".

Description

Description

If the current speed deviates from the set speed by less than the value of object 606D$_h$, bit 10 in object 6041$_h$ is set. The condition must be satisfied for longer than the time defined in object 606E (see also statusword in Profile Velocity Mode).

606Fh Velocity Threshold

Function

Speed in user-defined units above which the actual speed in Profile Velocity mode is considered to be nonzero.

Object description

Index	606F$_h$
Object name	Velocity Threshold
Object Code	VARIABLE
Data type	UNSIGNED16
Savable	yes, category: application
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	0000$_h$
Firmware version	FIR-v2013-B726332
Change history	

Description

If the actual speed is greater than the value in 606F$_h$(Velocity Threshold) for a time of 6070$_h$(Velocity Threshold Time), bit 12 in 6041$_h$(Statusword) has the value "0". The bit otherwise remains set to "1".

6070h Velocity Threshold Time

Function

Time in milliseconds above which an actual speed greater than the value in $606F_h$ in Profile Velocity mode is considered to be nonzero.

Object description

Index	6070_h
Object name	Velocity Threshold Time
Object Code	VARIABLE
Data type	UNSIGNED16
Savable	yes, category: application
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	0000_h
Firmware version	FIR-v2013-B726332
Change history	

Description

If the actual speed is greater than the value in $606F_h$(Velocity Threshold) for a time of 6070_h(Velocity Threshold Time), bit 12 in 6041_h(Statusword) has the value "0". The bit otherwise remains set to "1".

6071h Target Torque

Function

This object contains the target torque for the Profile Torque and Cyclic Synchronous Torque modes in tenths of a percent of the rated torque.

Object description

Index	6071_h
Object name	Target Torque
Object Code	VARIABLE
Data type	INTEGER16
Savable	yes, category: application
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	$000A_h$
Firmware version	FIR-v1426
Change history	Firmware version FIR-v1626: "Savable" entry changed from "no" to "yes, category: application".

Description

This object is calculated as thousandths of the torque, e.g., the value "500" means "50%" of the rated torque; "1100" is equivalent to 110%. The rated torque corresponds to the rated current in object $203B_h$:01.

The minimum of 6073_h and 6072_h is used as limit for the torque in 6071_h.

The target torque may not exceed the peak torque (proportional to the maximum motor current in 2031_h).

6072h Max Torque

Function

The object describes the maximum torque for the Profile Torque and Cyclic Synchronous Torque modes in tenths of a percent of the rated torque.

Object description

Index	6072_h
Object name	Max Torque
Object Code	VARIABLE
Data type	UNSIGNED16
Savable	yes, category: application
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	0064_h
Firmware version	FIR-v1426
Change history	

Description

This object is calculated as thousandths of the torque, e.g., the value "500" means "50%" of the rated torque; "1100" is equivalent to 110%. The rated torque corresponds to the rated current in object $203B_h$:01.

The minimum of 6073_h and 6072_h is used as limit for the torque in 6071_h.

The target torque may not exceed the peak torque (proportional to the maximum motor current in 2031_h).

6073h Max Current

Function

Contains the maximum current in tenths of a percent of the set rated current. Is limited by the maximum motor current (2031_h). See also I2t Motor overload protection.

NOTICE

For stepper motors, only the rated current is specified, not a maximum current. Therefore, the value of 6073_h should generally not exceed the value 1000 (100%).

Object description

Index	6073_h
Object name	Max Current

Object Code	VARIABLE
Data type	UNSIGNED16
Savable	yes, category: drive
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	$03E8_h$
Firmware version	FIR-v1825-B577172
Change history	

Description

The maximum current is calculated in tenths of a percent of the rated current as follows:

(6073_h*$203B_h$:01)/1000

The maximum current determines:

- the maximum current for the I2t Motor overload protection
- the rated current in *open-loop* mode.

6074h Torque Demand

Function

Current torque set value requested by the ramp generator in tenths of a percent of the rated torque for the internal controller.

Object description

Index	6074_h
Object name	Torque Demand
Object Code	VARIABLE
Data type	INTEGER16
Savable	no
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	0000_h
Firmware version	FIR-v1426
Change history	

Description

This object is calculated as thousandths of the torque, e.g., the value "500" means "50%" of the rated torque; "1100" is equivalent to 110%. The rated torque corresponds to the rated current in object $203B_h$:01.

The minimum of 6073_h and 6072_h is used as limit for the torque in 6071_h.

The target torque may not exceed the peak torque (proportional to the maximum motor current in 2031_h).

6075h Motor Rated Current

Function

Contains the rated current entered in $203B_h$:01_h in mA.

6077h Torque Actual Value

Function

This object indicates the current torque value in tenths of a percent of the rated torque for the internal controller.

Object description

Index	6077_h
Object name	Torque Actual Value
Object Code	VARIABLE
Data type	INTEGER16
Savable	no
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	0000_h
Firmware version	FIR-v1540
Change history	

Description

This object is calculated as thousandths of the torque, e.g., the value "500" means "50%" of the rated torque; "1100" is equivalent to 110%. The rated torque corresponds to the rated current in object $203B_h$:01.

The minimum of 6073_h and 6072_h is used as limit for the torque in 6071_h.

The target torque may not exceed the peak torque (proportional to the maximum motor current in 2031_h).

607Ah Target Position

Function

This object specifies the target position in user-defined units for the Profile Position and Cyclic Synchronous Position modes.

Object description

Index	$607A_h$
Object name	Target Position
Object Code	VARIABLE
Data type	INTEGER32
Savable	yes, category: application
Access	read / write
PDO mapping	RX-PDO
Allowed values	

Preset value	$00000E10_h$
Firmware version	FIR-v1426
Change history	Firmware version FIR-v1626: "Savable" entry changed from "no" to "yes, category: application".

607Bh Position Range Limit

Function

Contains the minimum and maximum position in user-defined units.

Object description

Index	$607B_h$
Object name	Position Range Limit
Object Code	ARRAY
Data type	INTEGER32
Savable	yes, category: application
Firmware version	FIR-v1426
Change history	

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	02_h

Subindex	01_h
Name	Min Position Range Limit
Data type	INTEGER32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000000_h

Subindex	02_h
Name	Max Position Range Limit
Data type	INTEGER32
Access	read / write
PDO mapping	RX-PDO
Allowed values	

Preset value	00000000_h

Description

If this range is exceeded or not reached, an overflow occurs. To prevent this overflow, limit values for the target position can be set in object $607D_h$ ("Software Position Limit").

607Ch Home Offset

Function

Specifies the difference between the zero position of the controller and the reference point of the machine in user-defined units.

Object description

Index	$607C_h$
Object name	Home Offset
Object Code	VARIABLE
Data type	INTEGER32
Savable	yes, category: application
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000000_h
Firmware version	FIR-v1426
Change history	

607Dh Software Position Limit

Function

Defines the limit positions relative to the reference point of the application in user-defined units.

Object description

Index	$607D_h$
Object name	Software Position Limit
Object Code	ARRAY
Data type	INTEGER32
Savable	yes, category: application
Firmware version	FIR-v1426
Change history	

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8

Access	read only
PDO mapping	no
Allowed values	
Preset value	02_h

Subindex	01_h
Name	Min Position Limit
Data type	INTEGER32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000000_h

Subindex	02_h
Name	Max Position Limit
Data type	INTEGER32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000000_h

Description

The absolute target position must lie within the limits set here. The Home Offset ($607C_h$) is not taken into account.

607Eh Polarity

Function

With this object, the direction of rotation can be reversed.

Object description

Index	$607E_h$
Object name	Polarity
Object Code	VARIABLE
Data type	UNSIGNED8
Savable	yes, category: application
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00_h
Firmware version	FIR-v1426
Change history	Firmware version FIR-v1738-B501312: "PDO mapping" table entry for subindex 00 changed from "no" to "RX-PDO".

Description

The following generally applies for direction reversal: If a bit is set to the value "1", reversal is activated. If the value is "0", the direction of rotation is as described in the respective mode.

7	6	5	4	3	2	1	0
POS	VEL						

VEL (Velocity)

Direction of rotation reversal in the following modes:

- Profile Velocity Mode
- Cyclic Synchronous Velocity Mode

POS (Position)

Direction of rotation reversal in the following modes:

- Profile Position Mode
- Cyclic Synchronous Position Mode

TIP

You can force an inversion of the rotary field that affects all operating modes. See object $3212_h:02_h$.

607Fh Max Profile Velocity

Function

Specifies the maximum speed in user-defined units for which the Mod i Profile Position and Profile Velocity.

Object description

Index	$607F_h$
Object name	Max Profile Velocity
Object Code	VARIABLE
Data type	UNSIGNED32
Savable	yes, category: application
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00007530_h
Firmware version	FIR-v1540
Change history	Firmware version FIR-v1738-B501312: "Object Name" entry changed from "Max profile velocity" to "Max Profile Velocity". Firmware version FIR-v1738-B501312: "Data type" entry changed from "INTEGER16" to "UNSIGNED32". Firmware version FIR-v1738-B501312: "Savable" entry changed from "no" to "yes, category: application". Firmware version FIR-v1738-B501312: "Access" table entry for subindex 00 changed from "read only" to "read/write".

	Firmware version FIR-v1738-B501312: "PDO mapping" table entry for subindex 00 changed from "TX-PDO" to "RX-PDO".

6080h Max Motor Speed

Function

Contains the maximum permissible speed of the motor in user-defined units.

Object description

Index	6080_h
Object name	Max Motor Speed
Object Code	VARIABLE
Data type	UNSIGNED32
Savable	yes, category: drive
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00001194_h
Firmware version	FIR-v1426
Change history	Firmware version FIR-v1614: "Savable" entry changed from "yes, category: application" to "yes, category: tuning". Firmware version FIR-v1738-B501312: "Object Name" entry changed from "Maximum Speed" to "Max Motor Speed". Firmware version FIR-v1738-B501312: "PDO mapping" table entry for subindex 00 changed from "no" to "RX-PDO". Firmware version FIR-v1748-B538662: "Savable" entry changed from "yes, category: tuning" to "yes, category: movement". Firmware version FIR-v1825-B577172: "Savable" entry changed from "yes, category: movement" to "yes, category: tuning". Firmware version FIR-v1825-B577172: "Savable" entry changed from "yes, category: tuning" to "yes, category: movement".

6081h Profile Velocity

Function

Specifies the maximum travel speed in user-defined units.

Object description

Index	6081_h
Object name	Profile Velocity
Object Code	VARIABLE
Data type	UNSIGNED32
Savable	yes, category: application
Access	read / write

PDO mapping	RX-PDO
Allowed values	
Preset value	000001F4$_h$
Firmware version	FIR-v1426
Change history	

6082h End Velocity

Function

Specifies the speed at the end of the traveled ramp in user-defined units.

Object description

Index	6082$_h$
Object name	End Velocity
Object Code	VARIABLE
Data type	UNSIGNED32
Savable	yes, category: application
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000000$_h$
Firmware version	FIR-v1426
Change history	

6083h Profile Acceleration

Function

Specifies the maximum acceleration in user-defined units.

Object description

Index	6083$_h$
Object name	Profile Acceleration
Object Code	VARIABLE
Data type	UNSIGNED32
Savable	yes, category: application
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	000001F4$_h$
Firmware version	FIR-v1426
Change history	

6084h Profile Deceleration

Function

Specifies the maximum deceleration (deceleration ramp) in user-defined units. Is limited by $60C6_h$.

Object description

Index	6084_h
Object name	Profile Deceleration
Object Code	VARIABLE
Data type	UNSIGNED32
Savable	yes, category: application
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	$000001F4_h$
Firmware version	FIR-v1426
Change history	

6085h Quick Stop Deceleration

Function

Specifies the maximum Quick Stop Deceleration in user-defined units. Depending on the operating mode, is limited by $60C6_h$ (Max Deceleration) and, if applicable, $60A4_h$ (Profile Jerk).

Object description

Index	6085_h
Object name	Quick Stop Deceleration
Object Code	VARIABLE
Data type	UNSIGNED32
Savable	yes, category: application
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00001388_h
Firmware version	FIR-v1426
Change history	

6086h Motion Profile Type

Function

Specifies the ramp type for the Profile Position and Profile Velocity modes.

Object description

Index	6086_h

Object name	Motion Profile Type
Object Code	VARIABLE
Data type	INTEGER16
Savable	yes, category: application
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	0000_h
Firmware version	FIR-v1426
Change history	

Description

Value = "0": = Trapezoidal ramp

Value = "3": Ramp with limited jerk

6087h Torque Slope

Function

This object contains the slope of the torque in Torque mode.

Object description

Index	6087_h
Object name	Torque Slope
Object Code	VARIABLE
Data type	UNSIGNED32
Savable	yes, category: application
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	$000003E8_h$
Firmware version	FIR-v1426
Change history	

Description

This object is calculated as thousandths of the torque, e.g., the value "500" means "50%" of the rated torque; "1100" is equivalent to 110%. The rated torque corresponds to the rated current in object $203B_h$:01.

The minimum of 6073_h and 6072_h is used as limit for the torque in 6071_h.

The target torque may not exceed the peak torque (proportional to the maximum motor current in 2031_h).

608Fh Position Encoder Resolution

Function

Contains the physical resolution (see objects $60E6_h$/ $60EB_h$) of the encoder/sensor that is used for position control (see 3203h Feedback Selection).

Object description

Index	$608F_h$
Object name	Position Encoder Resolution
Object Code	ARRAY
Data type	INTEGER32
Savable	yes, category: tuning
Firmware version	FIR-v1426
Change history	Firmware version FIR-v1738-B501312: "Savable" entry changed from "yes, category: application" to "yes, category: tuning". Firmware version FIR-v1738-B501312: "PDO mapping" table entry for subindex 01 changed from "no" to "RX-PDO". Firmware version FIR-v1738-B501312: "PDO mapping" table entry for subindex 02 changed from "no" to "RX-PDO". Firmware version FIR-v1748-B538662: "Data type" entry changed from "UNSIGNED32" to "INTEGER32".

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	02_h

Subindex	01_h
Name	Encoder Increments
Data type	INTEGER32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000000_h

Subindex	02_h
Name	Motor Revolutions
Data type	INTEGER32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000001_h

Description

Position Encoder Resolution = Encoder Increments (608F$_h$:01$_h$) / Motor Revolutions (608F$_h$:02$_h$)

6090h Velocity Encoder Resolution

Function

Contains the physical resolution (see objects 60E6$_h$/ 60EB$_h$) of the encoder/sensor that is used for speed control (see 3203h Feedback Selection).

Object description

Index	6090$_h$
Object name	Velocity Encoder Resolution
Object Code	ARRAY
Data type	INTEGER32
Savable	yes, category: tuning
Access	read only
PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v1738-B501312
Change history	Firmware version FIR-v1748-B538662: "Data type" entry changed from "UNSIGNED32" to "INTEGER32". Firmware version FIR-v1748-B538662: "Data type" entry changed from "UNSIGNED32" to "INTEGER32". Firmware version FIR-v1748-B538662: "Data type" entry changed from "UNSIGNED32" to "INTEGER32". Firmware version FIR-v1825-B577172: "Data type" entry changed from "INTEGER32" to "UNSIGNED32". Firmware version FIR-v1825-B577172: "Data type" entry changed from "INTEGER32" to "UNSIGNED32". Firmware version FIR-v1825-B577172: "Data type" entry changed from "INTEGER32" to "UNSIGNED32". Firmware version FIR-v1825-B577172: "Data type" entry changed from "UNSIGNED32" to "INTEGER32". Firmware version FIR-v1825-B577172: "Data type" entry changed from "UNSIGNED32" to "INTEGER32". Firmware version FIR-v1825-B577172: "Data type" entry changed from "UNSIGNED32" to "INTEGER32".

Value description

Subindex	00$_h$
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only

PDO mapping	no
Allowed values	
Preset value	02_h

Subindex	01_h
Name	Encoder Increments Per Second
Data type	INTEGER32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000000_h

Subindex	02_h
Name	Motor Revolutions Per Second
Data type	INTEGER32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000001_h

Description

Velocity Encoder Resolution = Encoder Increments per second (6090_h:01_h) / Motor Revolutions per second (6090_h:02_h)

6091h Gear Ratio

Function

Contains the gear ratio (number of motor revolutions per revolution of the output shaft) of the encoder/sensor that is used for position control (see 3203h Feedback Selection).

Object description

Index	6091_h
Object name	Gear Ratio
Object Code	ARRAY
Data type	UNSIGNED32
Savable	yes, category: application
Firmware version	FIR-v1426
Change history	Firmware version FIR-v1738-B501312: "PDO mapping" table entry for subindex 01 changed from "no" to "RX-PDO". Firmware version FIR-v1738-B501312: "PDO mapping" table entry for subindex 02 changed from "no" to "RX-PDO".

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	02_h

Subindex	01_h
Name	Motor Revolutions
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000001_h

Subindex	02_h
Name	Shaft Revolutions
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000001_h

Description

Gear Ratio = Motor Revolutions (6091_h:01_h) / Shaft Revolutions (6091_h:02_h)

6092h Feed Constant

Function

Contains the feed constant (feed in user-defined units per revolution of the output shaft) of the encoder/ sensor that is used for position control (see 3203h Feedback Selection).

Object description

Index	6092_h
Object name	Feed Constant
Object Code	ARRAY
Data type	UNSIGNED32
Savable	yes, category: application
Firmware version	FIR-v1426
Change history	

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	02_h

Subindex	01_h
Name	Feed
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000001_h

Subindex	02_h
Name	Shaft Revolutions
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000001_h

Description

Feed Constant = Feed (6092_h:01_h) / Shaft Revolutions (6092_h:02_h)

6096h Velocity Factor

Function

This object contains the factor that is used for converting from user-defined speed units. See chapter User-defined units.

Object description

Index	6096_h
Object name	Velocity Factor
Object Code	ARRAY
Data type	UNSIGNED32
Savable	yes, category: application
Access	read only
PDO mapping	no
Allowed values	

Preset value	
Firmware version	FIR-v1738-B501312
Change history	

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	02_h

Subindex	01_h
Name	Numerator
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000001_h

Subindex	02_h
Name	Divisor
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000001_h

Description

The subindices have the following functions:

- 01_h: Numerator of the factor
- 02_h: Denominator of the factor

6097h Acceleration Factor

Function

This object contains the factor that is used for converting from user-defined acceleration units. See chapter User-defined units.

Object description

Index	6097_h
Object name	Acceleration Factor

Object Code	ARRAY
Data type	UNSIGNED32
Savable	yes, category: application
Access	read only
PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v1738-B501312
Change history	

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	02_h

Subindex	01_h
Name	Numerator
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000001_h

Subindex	02_h
Name	Divisor
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000001_h

Description

The subindices have the following functions:

- 01_h: Numerator of the factor
- 02_h: Denominator of the factor

6098h Homing Method

Function

This object defines the Homing method in Homing mode.

Object description

Index	6098_h
Object name	Homing Method
Object Code	VARIABLE
Data type	INTEGER8
Savable	yes, category: application
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	23_h
Firmware version	FIR-v1426
Change history	

6099h Homing Speed

Function

Specifies the speeds for homing mode (6098_h) in user-defined units.

Object description

Index	6099_h
Object name	Homing Speed
Object Code	ARRAY
Data type	UNSIGNED32
Savable	yes, category: application
Firmware version	FIR-v1426
Change history	

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	02_h

Subindex	01_h

Name	Speed During Search For Switch
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000032_h

Subindex	02_h
Name	Speed During Search For Zero
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000003_h

Description

The speed for the search for the switch is specified in subindex 1.

The (lower) speed for the search for the reference position is specified in subindex 2.

NOTICE

- The speed in subindex 2 is simultaneously the initial speed when starting the acceleration ramp. If this is set too high, the motor loses steps or fails to turn at all.
  If the setting is too high, the index marking will be overlooked, especially with high-resolution encoders. The minimum detectable width of the index pulse is 31.25 µs.
  .
- The speed in subindex 1 must be greater than the speed in subindex 2.

609Ah Homing Acceleration

Function

Specifies the acceleration ramp for homing mode in user-defined units.

Object description

Index	$609A_h$
Object name	Homing Acceleration
Object Code	VARIABLE
Data type	UNSIGNED32
Savable	yes, category: application
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	$000001F4_h$
Firmware version	FIR-v1426
Change history	

Description

The ramp is only used when starting up. When the switch is reached, the motor immediately switches to the lower speed; when the end position is reached, it immediately stops.

60A2h Jerk Factor

Function

This object contains the factor that is used for converting from user-defined jerk units. See chapter User-defined units.

Object description

Index	$60A2_h$
Object name	Jerk Factor
Object Code	ARRAY
Data type	UNSIGNED32
Savable	yes, category: application
Access	read only
PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v1738-B501312
Change history	

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	02_h

Subindex	01_h
Name	Numerator
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000001_h

Subindex	02_h
Name	Divisor
Data type	UNSIGNED32
Access	read / write

PDO mapping	RX-PDO
Allowed values	
Preset value	00000001_h

Description

The subindices have the following functions:

- 01_h: Numerator of the factor
- 02_h: Denominator of the factor

60A4h Profile Jerk

Function

In the case of a ramp with limited jerk, the size of the jerk in user-defined units can be entered in this object. An entry with the value "0" means that the jerk is not limited.

Object description

Index	$60A4_h$
Object name	Profile Jerk
Object Code	ARRAY
Data type	UNSIGNED32
Savable	yes, category: application
Firmware version	FIR-v1426
Change history	Firmware version FIR-v1614: "Name" entry changed from "End Acceleration Jerk" to "Begin Deceleration Jerk". Firmware version FIR-v1614: "Name" entry changed from "Begin Deceleration Jerk" to "End Acceleration Jerk".

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	04_h

Subindex	01_h
Name	Begin Acceleration Jerk
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	$000003E8_h$

Subindex	02_h
Name	Begin Deceleration Jerk
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	$000003E8_h$

Subindex	03_h
Name	End Acceleration Jerk
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	$000003E8_h$

Subindex	04_h
Name	End Deceleration Jerk
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	$000003E8_h$

Description

- Subindex 01_h (*Begin Acceleration Jerk*): Initial jerk during acceleration
- Subindex 02_h (*Begin Deceleration Jerk*): Initial jerk during braking
- Subindex 03_h (*End Acceleration Jerk*): Final jerk during acceleration
- Subindex 04_h (*End Deceleration Jerk*): Final jerk during braking

60A8h SI Unit Position

Function

This object contains the position unit. See chapter User-defined units.

Object description

Index	$60A8_h$
Object name	SI Unit Position
Object Code	VARIABLE
Data type	UNSIGNED32
Savable	yes, category: application
Access	read / write
PDO mapping	no
Allowed values	

Preset value	FF410000$_h$
Firmware version	FIR-v1738-B501312
Change history	

Description

Object 60A8$_h$ contains:

- Bits 16 to 23: The position unit (see chapter Units)
- Bits 24 to 31: The exponent of a power of ten (see chapter Units)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Factor								Unit							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reserved (00h)								reserved (00h)							

60A9h SI Unit Velocity

Function

This object contains the speed unit. See chapter User-defined units.

Object description

Index	60A9$_h$
Object name	SI Unit Velocity
Object Code	VARIABLE
Data type	UNSIGNED32
Savable	yes, category: application
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00B44700$_h$
Firmware version	FIR-v1738-B501312
Change history	

Description

Object 60A9$_h$ contains:

- Bits 8 to 15: The time unit (see chapter Units)
- Bits 16 to 23: The position unit (see chapter Units)
- Bits 24 to 31: The exponent of a power of ten (see chapter Units)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Factor								Nominator (Position)							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Denominator (Time)								reserved (00h)							

60B0h Position Offset

Function

Offset for the position set value in user-defined units. Is taken into account in mode Cyclic Synchronous Position.

Object description

Index	$60B0_h$
Object name	Position Offset
Object Code	VARIABLE
Data type	INTEGER32
Savable	yes, category: application
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000000_h
Firmware version	FIR-v1738-B505321
Change history	

60B1h Velocity Offset

Function

Offset for the speed set value in user-defined units. Is taken into account in the Cyclic Synchronous Position, Cyclic Synchronous Velocity and Clock-direction mode modes.

Object description

Index	$60B1_h$
Object name	Velocity Offset
Object Code	VARIABLE
Data type	INTEGER32
Savable	yes, category: application
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000000_h
Firmware version	FIR-v1738-B505321
Change history	

60B2h Torque Offset

Function

Offset for the torque set value in tenths of a percent. Is taken into account in the Cyclic Synchronous Position, Cyclic Synchronous Velocity, Cyclic Synchronous Torque and Clock-direction mode modes.

Object description

Index	$60B2_h$
Object name	Torque Offset
Object Code	VARIABLE
Data type	INTEGER16
Savable	yes, category: application
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	0000_h
Firmware version	FIR-v1738-B505321
Change history	

60C1h Interpolation Data Record

Function

This object contains the demand position in user-defined units for the interpolation algorithm for the interpolated position operating mode.

Object description

Index	$60C1_h$
Object name	Interpolation Data Record
Object Code	ARRAY
Data type	INTEGER32
Savable	yes, category: application
Access	read only
PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v1512
Change history	Firmware version FIR-v1626: "Savable" entry changed from "no" to "yes, category: application".

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	01_h

Subindex	01_h
Name	1st Set-point
Data type	INTEGER32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000000_h

Description

The value is taken over at the next synchronization time.

60C2h Interpolation Time Period

Function

This object contains the interpolation time.

Object description

Index	$60C2_h$
Object name	Interpolation Time Period
Object Code	RECORD
Data type	INTERPOLATION_TIME_PERIOD
Savable	yes, category: application
Access	read only
PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v1426
Change history	

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	02_h

Subindex	01_h
Name	Interpolation Time Period Value
Data type	UNSIGNED8
Access	read / write
PDO mapping	no

Allowed values	
Preset value	01_h

Subindex	02_h
Name	Interpolation Time Index
Data type	INTEGER8
Access	read / write
PDO mapping	no
Allowed values	
Preset value	FD_h

Description

The subindices have the following functions:

- 01_h: Interpolation time.
- 02_h: Power of ten of the interpolation time: must have the value -3 (corresponds to the time basis in milliseconds).

The following applies here: cycle time = value of $60C2_h$:01_h * 10 $^{\text{value of 60C2:02}}$ seconds.

60C4h Interpolation Data Configuration

Function

This object offers the maximum buffer size, specifies the configured buffer organization of the interpolated data and offers objects for defining the size of the record and for deleting the buffer.

It is also used to store the position of other data points.

Object description

Index	$60C4_h$
Object name	Interpolation Data Configuration
Object Code	RECORD
Data type	INTERPOLATION_DATA_CONFIGURATION
Savable	yes, category: application
Access	read only
PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v1512
Change history	Firmware version FIR-v1540: "Access" table entry for subindex 05 changed from "read/write" to "write only". Firmware version FIR-v1540: "Access" table entry for subindex 06 changed from "read/write" to "write only". Firmware version FIR-v1626: "Savable" entry changed from "no" to "yes, category: application". Firmware version FIR-v1650-B472161: "Access" table entry for subindex 01 changed from "read/write" to "read only".

	Firmware version FIR-v2412-B1057638: entry "Name" changed from "Buffer Organization" to "Buffer Organisation".

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	06_h

Subindex	01_h
Name	Maximum Buffer Size
Data type	UNSIGNED32
Access	read only
PDO mapping	no
Allowed values	
Preset value	00000000_h

Subindex	02_h
Name	Actual Buffer Size
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000_h

Subindex	03_h
Name	Buffer Organisation
Data type	UNSIGNED8
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00_h

Subindex	04_h
Name	Buffer Position
Data type	UNSIGNED16
Access	read / write
PDO mapping	no
Allowed values	

Preset value	0000_h

Subindex	05_h
Name	Size Of Data Record
Data type	UNSIGNED8
Access	write only
PDO mapping	no
Allowed values	
Preset value	01_h

Subindex	06_h
Name	Buffer Clear
Data type	UNSIGNED8
Access	write only
PDO mapping	no
Allowed values	
Preset value	00_h

Description

The value of subindex 01_h contains the maximum possible number of interpolated records.

The value of subindex 02_h contains the current number of interpolated records.

If subindex 03_h is "00_h", this means a FIFO buffer organization; if it is "01_h", it specifies a ring buffer organization.

The value of subindex 04_h is unitless and specifies the next free buffer entry point.

The value of subindex 05_h is specified in units of "byte".

If the value "00_h" is written in subindex 06_h, it deletes the received data in the buffer, deactivates access and deletes all interpolated records.

If the value "01_h" is written in subindex 06_h, it activates access to the input buffer.

60C5h Max Acceleration

Function

This object contains the maximum permissible acceleration for the Profile Position and Profile Velocity modes.

Object description

Index	$60C5_h$
Object name	Max Acceleration
Object Code	VARIABLE
Data type	UNSIGNED32
Savable	yes, category: application
Access	read / write
PDO mapping	RX-PDO
Allowed values	

Preset value	00001388_h
Firmware version	FIR-v1426
Change history	

60C6h Max Deceleration

Function

This object contains the maximum permissible deceleration (deceleration ramp) for the Profile Position, Profile Velocity and Interpolated Position Mode operating modes.

Object description

Index	$60C6_h$
Object name	Max Deceleration
Object Code	VARIABLE
Data type	UNSIGNED32
Savable	yes, category: application
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00001388_h
Firmware version	FIR-v1426
Change history	

60E4h Additional Position Actual Value

Function

Contains the current actual position of all existing feedbacks in user-defined units.

Object description

Index	$60E4_h$
Object name	Additional Position Actual Value
Object Code	ARRAY
Data type	INTEGER32
Savable	no
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	
Firmware version	FIR-v1738-B501312
Change history	Firmware version FIR-v1748-B538662: "Data type" entry changed from "UNSIGNED32" to "INTEGER32". Firmware version FIR-v1748-B538662: "Data type" entry changed from "UNSIGNED32" to "INTEGER32".

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	02_h

Subindex	01_h - 02_h
Name	Additional Position Actual Value #1 - #2
Data type	INTEGER32
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	00000000_h

Description

The subindices have the following function:

- 00_h: Value="1" to "n", where "n" is the number of existing feedbacks.
- n_h:
  Subindex n contains the current actual position of the corresponding feedback.
  Subindex 01_h always corresponds to the first (and always existing) *sensorless* feedback.

60E5h Additional Velocity Actual Value

Function

Contains the current actual speed of all existing feedbacks in user-defined units.

Object description

Index	$60E5_h$
Object name	Additional Velocity Actual Value
Object Code	ARRAY
Data type	INTEGER32
Savable	no
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	
Firmware version	FIR-v1738-B501312
Change history	

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	02_h

Subindex	01_h - 02_h
Name	Additional Velocity Actual Value #1 - #2
Data type	INTEGER32
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	00000000_h

Description

The subindices have the following function:

- 00_h: Value="1" to "n", where "n" is the number of existing feedbacks.
- n_h:
  Subindex n contains the current actual speed of the corresponding feedback.
  Subindex 01_h always corresponds to the first (and always existing) *sensorless* feedback.

60E6h Additional Position Encoder Resolution - Encoder Increments

Function

With this object and with $60EB_h$, the resolution of each existing feedback is calculated.

Object description

Index	$60E6_h$
Object name	Additional Position Encoder Resolution - Encoder Increments
Object Code	ARRAY
Data type	INTEGER32
Savable	yes, category: tuning
Access	read only
PDO mapping	RX-PDO
Allowed values	
Preset value	
Firmware version	FIR-v1748-B538662
Change history	

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	RX-PDO
Allowed values	
Preset value	02_h

Subindex	01_h - 02_h
Name	Additional Position Encoder Resolution - Encoder Increments Feedback Interface #1 - #2
Data type	INTEGER32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000000_h

Description

The subindices have the following function:

- 00_h: Value="1" to "n", where "n" is the number of existing feedbacks.
- n_h:
  Subindex n contains the number of increments of the corresponding feedback.
  Subindex 01_h always corresponds to the first (and always existing) *sensorless* feedback.

The resolution of feedback "n" is calculated as follows:

Position Encoder Resolution = Encoder Increments ($60E6_h$:01_h) / Motor Revolutions ($60EB_h$:02_h)

NOTICE

The value "0" in a subindex means that the respective feedback is not connected and is not used. Thus, it is possible, for Example, to switch off the sensorless function to save computing time. This can be helpful if a *NanoJ* program needs the computing time.

If a value is not equal to "0" in a subindex, the controller checks the corresponding sensor when switching on. In case of an error (signal not present, invalid configuration/state), the error bit is set in the statusword and an error code stored in object 1003h.

60E8h Additional Gear Ratio - Motor Shaft Revolutions

Function

In this object and in $60ED_h$, you can set the gear ratio of each existing feedback.

Object description

Index	$60E8_h$
Object name	Additional Gear Ratio - Motor Shaft Revolutions

Object Code	ARRAY
Data type	UNSIGNED32
Savable	yes, category: application
Access	read only
PDO mapping	RX-PDO
Allowed values	
Preset value	
Firmware version	FIR-v1738-B501312
Change history	

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	RX-PDO
Allowed values	
Preset value	02_h

Subindex	01_h - 02_h
Name	Additional Gear Ratio - Motor Shaft Revolutions Feedback Interface #1 - #2
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000001_h

Description

The subindices have the following function:

- 00_h: Value = "n", where "n" is the number of existing feedbacks.
- n_h: Subindex "n" contains the number of motor revolutions for the corresponding feedback. Subindex 01_h always corresponds to the first (and always existing) *sensorless* feedback.

The gear ratio of feedback "n" is calculated as follows:

Gear Ratio = Motor Shaft Revolutions ($60E8_h$:n_h) / Driving Shaft Revolutions ($60ED_h$:n_h)

60E9h Additional Feed Constant - Feed

Function

In this object and in $60EE_h$, you can set a feed constant for each existing feedback.

Object description

Index	$60E9_h$

Object name	Additional Feed Constant - Feed
Object Code	ARRAY
Data type	UNSIGNED32
Savable	yes, category: application
Access	read only
PDO mapping	RX-PDO
Allowed values	
Preset value	
Firmware version	FIR-v1738-B501312
Change history	

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	RX-PDO
Allowed values	
Preset value	02_h

Subindex	01_h - 02_h
Name	Additional Feed Constant - Feed Feedback Interface #1 - #2
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000001_h

Description

The subindices have the following function:

- 00_h: Value = "n", where "n" is the number of existing feedbacks.
- n_h: Subindex "n" contains the feed in user-defined units for the corresponding feedback. Subindex 01_h always corresponds to the first (and always existing) *sensorless* feedback.

The feed constant of feedback "n" is calculated as follows:

Feed Constant = Feed ($60E9_h$:n_h) / Driving Shaft Revolutions ($60EE_h$:n_h)

60EBh Additional Position Encoder Resolution - Motor Revolutions

Function

With this object and with $60E6_h$, the resolution of each existing feedback is calculated.

Object description

Index	$60EB_h$

Object name	Additional Position Encoder Resolution - Motor Revolutions
Object Code	ARRAY
Data type	UNSIGNED32
Savable	yes, category: tuning
Access	read only
PDO mapping	RX-PDO
Allowed values	
Preset value	
Firmware version	FIR-v1738-B501312
Change history	

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	RX-PDO
Allowed values	
Preset value	02_h

Subindex	01_h - 02_h
Name	Additional Position Encoder Resolution - Motor Revolutions Feedback Interface #1 - #2
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000001_h

Description

The subindices have the following function:

- 00_h: Value="1" to "n", where "n" is the number of existing feedbacks.
- n_h:
  Subindex n contains the number of motor revolutions of the corresponding feedback.
  Subindex 01_h always corresponds to the first (and always existing) *sensorless* feedback.

The resolution of feedback "n" is calculated as follows:

Position Encoder Resolution = Encoder Increments ($60E6_h$:n_h) / Motor Revolutions ($60EB_h$:n_h)

60EDh Additional Gear Ratio - Driving Shaft Revolutions

Function

In this object and in $60E8_h$, you can set the gear ratio of each existing feedback.

Object description

Index	$60ED_h$
Object name	Additional Gear Ratio - Driving Shaft Revolutions
Object Code	ARRAY
Data type	UNSIGNED32
Savable	yes, category: application
Access	read only
PDO mapping	RX-PDO
Allowed values	
Preset value	
Firmware version	FIR-v1738-B501312
Change history	

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	RX-PDO
Allowed values	
Preset value	02_h

Subindex	01_h - 02_h
Name	Additional Gear Ratio - Driving Shaft Revolutions Feedback Interface #1 - #2
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000001_h

Description

The subindices have the following function:

- 00_h: Value = "n", where "n" is the number of existing feedbacks.
- n_h: Subindex "n" contains the number of revolutions of the output shaft for the corresponding feedback. Subindex 01_h always corresponds to the first (and always existing) *sensorless* feedback.

The gear ratio of feedback "n" is calculated as follows:

Gear Ratio = Motor Shaft Revolutions ($60E8_h$:n_h) / Driving Shaft Revolutions ($60ED_h$:n_h)

60EEh Additional Feed Constant - Driving Shaft Revolutions

Function

In this object and in $60E9_h$, you can set a feed constant for each existing feedback.

Object description

Index	$60EE_h$
Object name	Additional Feed Constant - Driving Shaft Revolutions
Object Code	ARRAY
Data type	UNSIGNED32
Savable	yes, category: application
Access	read only
PDO mapping	RX-PDO
Allowed values	
Preset value	
Firmware version	FIR-v1738-B501312
Change history	

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	RX-PDO
Allowed values	
Preset value	02_h

Subindex	01_h - 02_h
Name	Additional Feed Constant - Driving Shaft Revolutions Feedback Interface #1 - #2
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000001_h

Description

The subindices have the following function:

- 00_h: Value = "n", where "n" is the number of existing feedbacks.
- n_h: Subindex "n" contains the number of revolutions of the output shaft for the corresponding feedback. Subindex 01_h always corresponds to the first (and always existing) *sensorless* feedback.

The feed constant of feedback "n" is calculated as follows:

Feed Constant = Feed ($60E9_h$:n_h) / Driving Shaft Revolutions ($60EE_h$:n_h)

60F2h Position Option Code

Function

The object describes the positioning behavior in Profile Position mode.

Object description

Index	$60F2_h$
Object name	Position Option Code
Object Code	VARIABLE
Data type	UNSIGNED16
Savable	yes, category: application
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	0000_h
Firmware version	FIR-v1446
Change history	Firmware version FIR-v1614: "Savable" entry changed from "no" to "yes, category: application". Firmware version FIR-v2412-B1057638: entry "Object Name" changed from "Positioning Option Code" to "Position Option Code".

Description

Only the following bits are supported at the present time:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MS	RESERVED [3]			IP OPTION [4]				RADO [2]		RRO [2]		CIO [2]		REL. OPT. [2]	

REL. OPT. (Relative Option)

These bits determine the behavior with relative rotating movement in "profile position" mode if bit 6 of controlword 6040_h = "1" is set.

Bit 1	Bit 0	Definition
0	0	Position movements are executed relative to the previous (internal absolute) target position (each relative to 0 if there is no previous target position)
0	1	Position movements are executed relative to the preset value (or output) of the ramp generator.
1	0	Position movements are performed relative to the current position (object 6064_h).
1	1	Reserved

RRO (Request-Response Option)

These bits determine the behavior when passing controlword 6040_h bit 4 ("new setpoint") – in this case, the controller releases the bit itself. This eliminates the need to externally reset the bit to "0" afterwards. After the bit is set to the value "0" by the controller, bit 12 ("setpoint acknowledgment") is also set to the value "0" in statusword 6041_h.

NOTICE

These options cause the controller to modify object controlword 6040_h.

Bit 5	Bit 4	Definition
0	0	The functionality is as described under Setting travel commands.
0	1	The controller releases the "new setpoint" bit as soon as the current targeted movement has reached its target.
1	0	The controller releases the "new setpoint" bit as soon this is possible for the controller.
1	1	Reserved

RADO (Rotary Axis Direction Option)

These bits determine the direction of rotation in "profile position" mode.

Bit 7	Bit 6	Definition
0	0	Normal positioning similar to a linear axis: If one of the "Position Range Limits" – $607B_h$:01_h and 02_h – is reached or exceeded, the preset is automatically transferred to the other end of the limit. Only with this bit combination is a movement greater than the modulo value possible.
0	1	Positioning only in negative direction: If the target position is greater than the current position, the axis moves to the target position via the "Min Position Range Limit" from object $607D_h$:01_h.
1	0	Positioning only in positive direction: If the target position is less than the current position, the axis moves to the target position via the "Max Position Range Limit" from object $607D_h$:01_h.
1	1	Positioning with the shortest distance to the target position. If the difference between the current position and the target position in a 360° system is less than 180°, the axis moves in the positive direction.

60F4h Following Error Actual Value

Function

This object contains the current following error in user-defined units.

Object description

Index	$60F4_h$
Object name	Following Error Actual Value
Object Code	VARIABLE
Data type	INTEGER32
Savable	no
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	00000000_h
Firmware version	FIR-v1426
Change history	

60F8h Max Slippage

Function

Defines the maximum allowed slippage error in user-defined units symmetrically to the set speed in Profile Velocity mode.

Object description

Index	$60F8_h$
Object name	Max Slippage
Object Code	VARIABLE
Data type	INTEGER32
Savable	yes, category: application
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000190_h
Firmware version	FIR-v1738-B501312
Change history	

Description

If the actual speed deviates so much from the set speed that the value (absolute value) of this object is exceeded, bit 13 in object 6041_h is set. The deviation must last longer than the time in object $203F_h$.

If the value of $60F8_h$ is set to "7FFFFFFF"$_h$, slippage error monitoring is switched off.

A reaction to the slippage error can be set in object 3700_h. If a reaction is defined, an error is also entered in object 1003_h.

60FAh Control Effort

Function

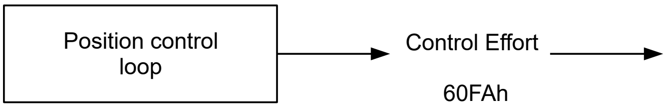
This object contains the correction speed (control variable) in user-defined units that is fed to the velocity controller by the position controller.

Object description

Index	$60FA_h$
Object name	Control Effort
Object Code	VARIABLE
Data type	INTEGER32
Savable	no
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	00000000_h
Firmware version	FIR-v1748-B531667
Change history	

Description

The position controller calculates a correction speed (in user-defined units) from the difference between the current position and the demand position which is then passed on to the velocity controller. This correction value is dependent on the proportional component and integral component of the position controller. See also chapter Closed-Loop.



60FCh Position Demand Internal Value

Function

Object description

Index	60FC$_h$
Object name	Position Demand Internal Value
Object Code	VARIABLE
Data type	INTEGER32
Savable	no
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	00000000$_h$
Firmware version	FIR-v1738-B501312
Change history	

60FDh Digital Inputs

Function

With this object, the digital inputs of the motor can be read.

Object description

Index	60FD$_h$
Object name	Digital Inputs
Object Code	VARIABLE
Data type	UNSIGNED32
Savable	no
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	00000000$_h$

Firmware version	FIR-v1426
Change history	

Description

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
							...	IN 7	IN 6	IN 5	IN 4	IN 3	IN 2	IN 1	IN 0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
												IL	HS	PLS	NLS

NLS (Negative Limit Switch)
Negative limit switch

PLS (Positive Limit Switch)
Positive limit switch

HS (Home Switch)
Home switch

IL (Interlock)
Interlock

IN n (Input n)
Input n – the number of used bits is dependent on the given controller.

60FEh Digital Outputs

Function

With this object, the digital outputs of the motor can be written.

Object description

Index	$60FE_h$
Object name	Digital Outputs
Object Code	ARRAY
Data type	UNSIGNED32
Savable	yes, category: application
Firmware version	FIR-v1426
Change history	Firmware version FIR-v1626: "Savable" entry changed from "no" to "yes, category: application".

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	

Preset value	01_h

Subindex	01_h
Name	Physical Outputs
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000000_h

Description

To write the outputs, the entries in object 3250_h, subindex 02_h to 05_h, must also be taken into account.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
											...	OUT4	OUT3	OUT2	OUT1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
															BRK

BRK (Brake)

Bit for the brake output (if the controller supports this function):

Value "1" means that the brake is activated (no current can flow between the two pins of the brake connection; the brake is closed).

OUT n (Output No n)

Bit for the respective digital output; the exact number of digital outputs is dependent on the controller.

60FFh Target Velocity

Function

In this object, the target speed for the profile velocity and cyclic synchronous velocity modes is entered in user-defined units.

Object description

Index	$60FF_h$
Object name	Target Velocity
Object Code	VARIABLE
Data type	INTEGER32
Savable	yes, category: application
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000000_h
Firmware version	FIR-v1426
Change history	Firmware version FIR-v1626: "Savable" entry changed from "no" to "yes, category: application".

6502h Supported Drive Modes

Function

The object describes the supported operating modes in object 6060_h.

Object description

Index	6502_h
Object name	Supported Drive Modes
Object Code	VARIABLE
Data type	UNSIGNED32
Savable	no
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	$000003EF_h$
Firmware version	FIR-v1426
Change history	

Description

The set bit specifies whether the respective mode is supported. If the value of the bit is "0", the mode is not supported.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
						CST	CSV	CSP	IP	HM		TQ	PV	VL	PP

PP
: Profile Position Mode

VL
: Velocity Mode

PV
: Profile Velocity Mode

TQ
: Torque Mode

HM
: Homing Mode

IP
: Interpolated Position Mode

CSP
: Cyclic Synchronous Position Mode

CSV
: Cyclic Synchronous Velocity Mode

CST

Cyclic Synchronous Torque Mode

6503h Drive Catalogue Number

Function

Contains the device name as character string.

Object description

Index	6503_h
Object name	Drive Catalogue Number
Object Code	VARIABLE
Data type	VISIBLE_STRING
Savable	no
Access	read only
PDO mapping	no
Allowed values	
Preset value	0
Firmware version	FIR-v1426
Change history	

6505h Http Drive Catalogue Address

Function

This object contains the manufacturer's web address as a character string.

Object description

Index	6505_h
Object name	Http Drive Catalogue Address
Object Code	VARIABLE
Data type	VISIBLE_STRING
Savable	no
Access	read only
PDO mapping	no
Allowed values	
Preset value	0
Firmware version	FIR-v1426
Change history	

11 Copyrights

11.1 Introduction

Integrated in the Nanotec software are components from products from external software manufacturers. In this chapter, you will find the copyright information regarding the used external software sources.

11.2 AES

FIPS-197 compliant AES implementation

Based on XySSL: Copyright (C) 2006-2008 Christophe Devine

Copyright (C) 2009 Paul Bakker <polarssl_maintainer at polarssl dot org>

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution; or, the application vendor's website must provide a copy of this notice.
- Neither the names of PolarSSL or XySSL nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

The AES block cipher was designed by Vincent Rijmen and Joan Daemen.

http://csrc.nist.gov/encryption/aes/rijndael/Rijndael.pdf

http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

11.3 MD5

MD5C.C - RSA Data Security, Inc., MD5 message-digest algorithm

Copyright (C) 1991-2, RSA Data Security, Inc. Created 1991. All rights reserved.

License to copy and use this software is granted provided that it is identified as the "RSA Data Security, Inc. MD5 Message-Digest Algorithm" in all material mentioning or referencing this software or this function.

License is also granted to make and use derivative works provided that such works are identified as "derived from the RSA Data Security, Inc. MD5 Message-Digest Algorithm" in all material mentioning or referencing the derived work.

RSA Data Security, Inc. makes no representations concerning either the merchantability of this software or the suitability of this software for any particular purpose. It is provided "as is" without express or implied warranty of any kind.

These notices must be retained in any copies of any part of this documentation and/or software.

11.4 uIP

11.5 DHCP

11.6 CMSIS DSP Software Library

11.7 FatFs

FatFs - FAT file system module include file R0.08 (C)ChaN, 2010

FatFs module is a generic FAT file system module for small embedded systems.

This is a free software that opened for education, research and commercial

developments under license policy of following trems.

Copyright (C) 2010, ChaN, all right reserved.

The FatFs module is a free software and there is NO WARRANTY.

No restriction on use. You can use, modify and redistribute it for

personal, non-profit or commercial product UNDER YOUR RESPONSIBILITY.

Redistributions of source code must retain the above copyright notice.

11.8 Protothreads

Protothread class and macros for lightweight, stackless threads in C++.

This was "ported" to C++ from Adam Dunkels' protothreads C library at: http://www.sics.se/~adam/pt/

Originally ported for use by Hamilton Jet (www.hamiltonjet.co.nz) by Ben Hoyt, but stripped down for public release. See his blog entry about it for more information: http://blog.micropledge.com/2008/07/protothreads/

Original BSD-style license

Copyright (c) 2004-2005, Swedish Institute of Computer Science.

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. Neither the name of the Institute nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

This software is provided by the Institute and contributors "as is" and any express or implied warranties, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose are disclaimed. In no event shall the Institute or contributors be liable for any direct, indirect, incidental, special, exemplary, or consequential damages (including, but not limited to, procurement of substitute goods or services; loss of use, data, or profits; or business interruption) however caused and on any theory of liability, whether in contract, strict liability, or tort (including negligence or otherwise) arising in any way out of the use of this software, even if advised of the possibility of such damage.

11.9 lwIP

Copyright (c) 2001-2004 Swedish Institute of Computer Science.

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. The name of the author may not be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE AUTHOR ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO

This file is part of the lwIP TCP/IP stack.

Author: Adam Dunkels <adam@sics.se>

11.10 littlefs

```
/*
* The little filesystem
*
* Copyright (c) 2017, Arm Limited. All rights reserved.
* SPDX-License-Identifier: BSD-3-Clause
*/
```